TAMPEREEN TEKNILLINEN YLIOPISTO

TEEMU KUMPUMÄKI

KAMERAJÄRJESTELMÄN JA KUVANKÄSITTELYMENETELMIEN KEHITTÄMINEN KALLIOSEINÄMÄN MUUTOSTEN TARKKAILUUN

Diplomityö

Tarkastaja: professori Tarmo Lipping
Tarkastaja ja aihe hyväksytty Tieto- ja sähkötekniikan tiedekunta-neuvoston kokouksessa 9.2.2011.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
KUMPUMÄKI, TEEMU: Kamerajärjestelmän sekä kuvankäsittelymenetelmien kehittäminen kallioseinämän muutosten tarkkailuun
Diplomityö, 46 sivua
Huhtikuu 2011
Pääaine: Signaalinkäsittely
Tarkastaja: professori Tarmo Lipping
Avainsanat: Kamerajärjestelmä, kuvien oikaisu, kuvien yhdistäminen

Diplomityössä toteutettiin automaattinen usean kameran kamerajärjestelmä pitkän aikavälin seurantaan. Lisäksi työssä kehitettiin kerätylle kuva-aineistolle käsittelymenetelmät lopputuloksen muodostamiseksi. Kerätty kuva-aineisto yhdistettiin ja käsiteltiin mahdollisimman hyvälaatuiseksi. Aikavälin yli kerätyt kuvat tallennettiin lopputuloksessa videotiedostoksi, josta niiden analysoiminen oli yksinkertaista.

Kamerajärjestelmä toteutettiin Posiva Oy:n tekemää tutkimusta varten. Kuva-aineistoa kerättiin POSE-tutkimuksen (Posiva's Olkiluoto Spalling Experiment) yhteydessä tukemaan muilla mittaustavoilla saatuja tuloksia. Kamerajärjestelmällä kuvattiin muutoksia kallion pinnassa, kun ympäröivän kallion jännitystila muuttui viereisen reiän porauksen ja lämmitysvaiheen aikana.

Kamerajärjestelmä toteutettiin kuva-aineistolle annettujen määrittelyiden perusteella, kuitenkin niin että järjestelmä oli kohtuulliseen hintaan toteutettavissa. Annettujen määrittelyiden perusteella päädyttiin kompromissiratkaisuun, jossa järjestelmän keräämä kuva-aineisto saavutti riittävän kattavuuden ja laadun.

Kuva-aineistoa kerättiin usealla kameralla vaikeahkosta paikasta, kallioon poratun ison tutkimusreiän sisältä, jolloin usean kameran kuvien yhdistämistä varten täytyi kuville laskea oikaisut ennen kuvien yhdistämistä. Kuvien oikaiseminen toteutettiin PC:n näytönohjaimen avulla tapahtuvalla laskennalla, jossa kuvien oikaiseminen toteutettiin kuvattavasta kohteesta tehdyn mallin ja tekstuuriprojektioiden avulla.

Lopputulokseksi saadun videon tarkoituksena on tallentaa kokeiden aikana tapahtuvat muutokset. Muutoksien näkemiseksi kuvat käsiteltiin mahdollisimman selkeään muotoon. Muutosten visuaalista havaitsemista varten kehitettiin sovellus, joka mahdollistaa eri aikoina otettujen kuvien välisen erotuksen laskemisen kuvasarjalle. Sovelluksen avulla pienimmätkin muutokset kuvien välillä ovat hyvin havaittavissa.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
KUMPUMÄKI, TEEMU: Developing a camera system and preprocessing methods for rock wall transition monitoring
Master of Science Thesis, 46 pages
April 2011
Major: Signal processing
Examiner: professor Tarmo Lipping
Keywords: Camera system, image shape correction, image composition

The first part of this Master's thesis consists of developing an automatic multi camera image acquisition system for long time range image acquisition. The second part of the thesis consists of developing image processing methods for generating results from the acquired images. Acquired images from the multiple cameras were merged and processed for optimum results. Acquired images over the time period were stored in the final result as a video.

Camera system was developed as a part of Posiva Oy's research. Image data was collected as a part of POSE-experiment (Posiva's Olkiluoto Spalling Experiment) project to support other measurement methods. During the research the camera system monitored structural changes in the rock wall while surrounding rock stresses were altered by boring the second hole and finally heating the rock mass.

The system was implemented within given specifications, while still maintaining reasonable price for the system. Some compromise solutions had to be made in the specification in order to complete the system. Image data coverage and quality were maximized for devices in the given price class.

Image data were collected with multiple cameras from a difficult position inside a large full face bored hole in the rock. When merging images taken from a non-planar surface, image distortions need to be corrected before images can be seamlessly merged. Image distortions were corrected using PC graphics card's computational power. Corrections were calculated with a model made from the drilled hole and method called texture projection.

The purpose of the video produced as a result of the image processing is to record the events that occur during different phases of experiment. For visual inspection of video data, to see the smallest transitions in the rock, images were processed to be as clear as possible. An application for showing transitions in the rock was developed to help visual analysis. This application enables to visualize differences between images taken at different times. Difference images show clearly even the smallest discrepancy between the images.

ALKUSANAT

Diplomityö tehtiin Tampereen teknillisen yliopiston Porin yksikössä pääosin vuoden 2010 aikana. Diplomityön tilaajana toimi Kimmo Kemppainen Posiva Oy:stä.

Haluan kiittää kaikkia projektiin osallistuneita henkilöitä, sekä erityisesti työn ohjaajina toimineita Kimmo Kemppaista Posiva Oy:stä, sekä Tarmo Lippingiä TTY:ltä. Kiitokset myös TTY:n Porin yksikön Petteri Kermiselle ja Jari Turuselle signaalinkäsittelyyn liittyvistä vinkeistä.

SISÄLLYS

LYHENTEET JA MERKINNÄT

API	Application Programming Interface, ohjelmakirjasto-rajapinta.
ASP	Advanced Simple Profile, MPEG-4 standardin määrittelemä videon pakkausprofiili.
C tai CS kiinnike	Pienikokoinen kameran linssityyppi, jossa kiinnitys kameraan kierteellä.
Cat-5	Parikaapelityyppi.
FOV	Field Of View, näkökenttä.
GLSL	OpenGL Shading Language, näytönohjaimen ohjaukseen tarkoitettu ohjelmointikieli.
HSV	Yksittäisten kuvapisteiden esittämiseen käytettävä väriavaruusmalli.
ffmpeg	Videotiedostojen käsittelyyn erikoistunut ohjelmakirjasto.
ImageMagick	Kuvankäsittelyalgoritmeista koostuva ohjelmakirjasto.
IDE	Integrated Development Environment, ohjelmistokehitys-ympäristö.
JPEG	Kuvanpakkausformaatti.
MPEG	Videonpakkausformaatti, useita eri versioita.
NAS	Network Attached Storage, verkkolevy.
OpenGL	Tietokonegrafiikkakirjasto.
ONKALO	Posiva Oy:n ydinjätteen loppusijoituksen tutkimusluola.
POSE-tutkimus	Posivan tutkimus, jossa tutkitaan kallion hilseilyn kehittymistä (Posiva’s Olkiluoto Spalling Experiment).
Ray tracing	Kuvangenerointi tapa, jossa simuloidaan valonsäteiden etenemistä ja kohtaamista virtuaalikappaleiden kanssa.
RAW	Kameran kuvakennon tuottama raakadataformaatti.
RGB	Värimalli, jossa värit sekoitetaan punaisesta, vihreästä ja sinisestä.
SDK	Software Development Kit, kokoelma ohjelmistokehitys-työkaluja.
Tekstuuri	Tietokonegrafiikassa käytetty nimi pintakuvioinnille.
Trussi	Alumiininen kehikkorakenne.
uEye	IDS Imaging Development Systems GmbH:n valmistama kameramallisto.
UPS	Uninterruptible Power Supply, varavirtajärjestelmä.
USB	Universal Serial Bus, sarjaväylä.

1 JOHDANTO

Diplomityön aiheena oli kamerajärjestelmän toteuttaminen osaksi POSE-tutkimuksen mittauksia ja kerätyn kuvamateriaalin käsitteleminen helposti tulkittavaan muotoon. Työn tilasi Kimmo Kemppainen Posiva Oy:stä. Posiva Oy on Teollisuuden Voima Oyj:n ja Fortum Power & Heat Oy:n omistama asiantuntijaorganisaatio, jonka tehtäväksi on asetettu omistajayhtiöiden käytetyn ydinpolttoaineen loppusijoituksen suunnittelu, loppusijoituslaitoksen rakentaminen, sekä laitoksen käyttö ja sulkeminen käytön jälkeen.

Kamerajärjestelmä on yksi POSE-tutkimuksen mittalaitteista, joiden tavoitteena on kerätä tietoa käytetyn ydinpolttoaineen turvallisen loppusijoituksen suunnittelua varten. Käytetyn polttoaineen loppusijoituksessa on otettava huomioon radioaktiivisen jätteen vaarallisuus erittäin pitkällä aikavälillä. Pitkien aikavälien ollessa kyseessä, on oltava riittävän varmoja valituista rakenteista ja ratkaisuista, ettei radioaktiivinen aine pääse luontoon mitään kautta. POSE-tutkimuksessa kerätään tietoja kiven hilseilylujuudesta ja kestävyydestä. Tuloksilla tarkennetaan aikaisemmin maailmalla tehtyjen tutkimusten tuloksia, joita käytetään suunnittelun lähtötietoina.

Diplomityöhön liittyvä projekti suoritettiin kahdessa osassa, aluksi oli tärkeää saada määritelmän mukainen kamerajärjestelmä toimintakuntoon, POSE-tutkimukseen liittyvien porauksien alkaessa. Kamerajärjestelmän asennuksen ja käyttöönoton jälkeen projektissa siirryttiin toiseen vaiheeseen, jossa kameroiden ottamille kuville laadittiin kuvankäsittelyohjelmisto, jonka avulla kameroiden ottamat kuvasarjat saadaan muutettua lopputulokseksi, työn tilaajan asettamien vaatimusten mukaiseen muotoon.

Tilatun kamerajärjestelmän toteuttaminen aloitettiin saatujen määrittelyjen perusteella helmikuussa 2010 ja kamerajärjestelmä otettiin käyttöön kesäkuussa 2010. Kuvien käsittelyyn tarkoitettu ohjelmisto toteutettiin kamerajärjestelmän valmistumisen jälkeen ja se saatiin valmiiksi alkukeväästä 2011.

Luvussa kaksi esitellään työlle annetut lähtökohdat, joiden perusteella kamerajärjestelmän kehittäminen aloitettiin ja kerrotaan minkälaiseen lopputulokseen työssä pitäisi päästä. Kolmannessa luvussa esitellään kuva-aineiston tallentamiseen ja käsittelyyn käytetyt algoritmit ja menetelmät. Neljännessä luvussa keskitytään työn toteutuksessa tehtyjen valintojen ja toimintatapojen esittelyyn. Viimeisessä luvussa arvioidaan työn tuloksia ja toteutuksen onnistumista.

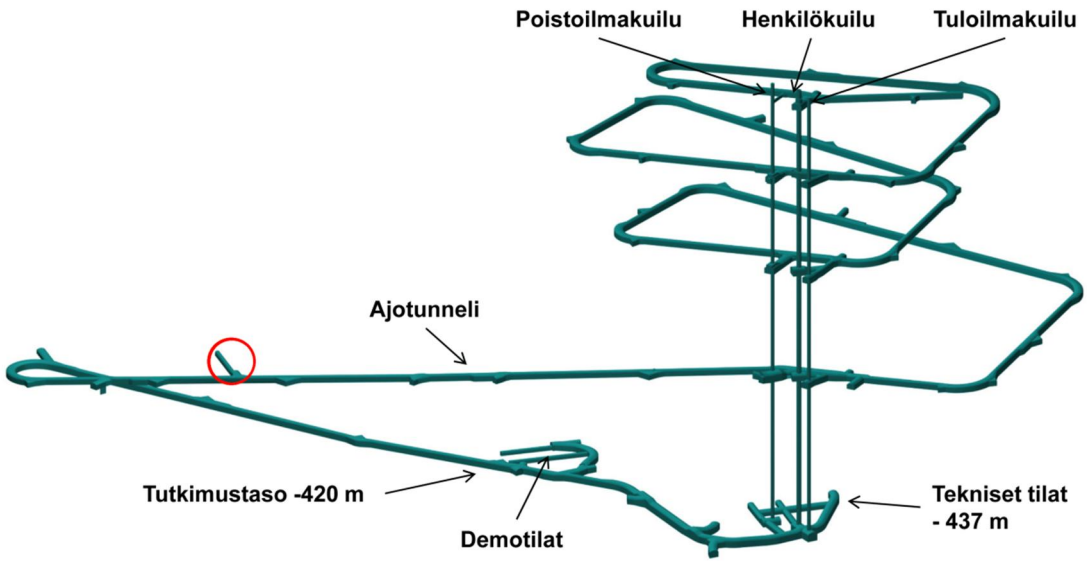
2 TYÖN TAUSTA

Tässä luvussa esitellään työn kannalta oleelliset yleiset tiedot toteutettavasta kamerajärjestelmästä, sekä minkälaisen lopputuloksen työn tilaaja on halunnut saavuttaa kuva-aineiston osalta. Aiheen käsittely lähtee työn toteutusympäristön esittelystä. Olosuhteiden esittelyn jälkeen perustellaan työssä toteutetun järjestelmän tarpeellisuutta ja roolia POSE-tutkimuksessa. Luvun lopuksi kerrotaan, millaiset vaatimukset työn lopputulokselle asetettiin.

2.1 ONKALO-tutkimustunneli

ONKALO-tutkimustunneli on Posiva Oy:n tutkimustunneli, jossa tehtävistä ydinjätteen loppusijoitukseen liittyvistä tutkimuksista saadaan tarkentavaa tietoa loppusijoituslaitoksen rakentamislupahakemusta varten. Tunnelissa suoritetaan tutkimuksia samalla, kun ajotunnelin louhiminen loppusijoitussyvyyteen etenee.

Tunnelissa tehdyillä tutkimuksilla ja teknisillä demonstraatioilla kerätään tietoa varsinaista ydinjätteen loppusijoitusta varten ja kokeillaan suunniteltuja teknisiä ratkaisuja joiden avulla loppusijoitus toteutetaan. Ajotunnelin louhinnan yhteydessä sen varrelle louhittiin erikokoisia tutkimustiloja, joissa suoritetaan loppusijoituksen kannalta oleellisia tutkimuksia.



Kuva 2.1. *ONKALO-tutkimustunnelin rakennekuva, tutkimustila 3 on merkitty punaisella ympyrällä, lähde: Posiva Oy.*

POSE-tutkimus tapahtuu tutkimustilassa 3, joka sijaitsee ONKALO:ssa noin 345 metrin syvyydessä, ajotunnelissa 3620 metrin kohdalla. Kuvassa 2.1 on

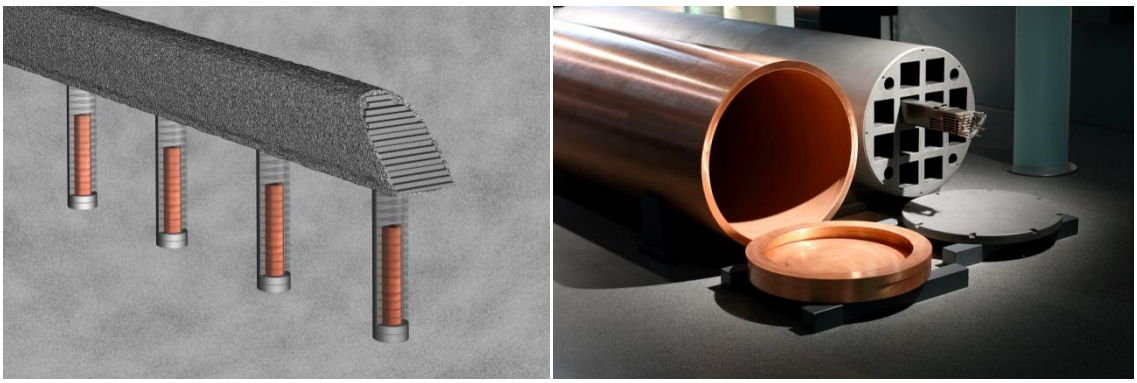
tutkimustunnelin rakennekuva, jossa POSE-tutkimustila on merkittynä kuvaan punaisella ympyrällä. Tutkimustilaan on mahdollista päästä ainoastaan autolla ajotunnelia pitkin.

2.2 POSE-tutkimus

POSE-tutkimuksen (Posiva's Olkiluoto Spalling Experiment) tarkoituksena on selvittää Olkiluodon alueella vallitsevien kivilajien lujuusominaisuuksia. Kivilajien mekaaniset ominaisuudet ja käyttäytyminen eri syvyyksissä on tärkeää selvittää loppusijoituksen kannalta. POSE-tutkimuksen aikana suoritetaan erilaisia mittauksia joissa kerätään tietoa kiven käyttäytymisestä tutkimuksen eri vaiheissa. Saaduilla tuloksilla täsmennetään maailmalta saatuja tutkimustuloksia, joita käytetään tunneleiden suunnittelun lähtötietoina. [1]

POSE-tutkimuksessa on tarkoituksena selvittää kiven hilseilylujuus alueella vallitsevissa jännitysolosuhteissa. Vastaavaa tutkimusta ei ole mahdollista suorittaa laboratoriomittakaavassa. Kiven hilseilylujuus on tärkeää selvittää, jotta pystytään paremmin ennustamaan millä tavalla kivi voi hilseillä päätunneleissa, loppusijoitustunneleissa ja loppusijoitusrei'issä. Saatuja tuloksia käytetään apuna ONKALO:n toteuttamattomien osien suunnittelussa sekä myöhemmin toteutettavan loppusijoitustunnelin suunnittelussa. [2] [3]

Loppusijoituksessa kuvan 2.2 oikean puolen mukaiset ydinjätekapselit on esitetty sijoitettavaksi kallioon porattuihin reikiin, kuvan vasemman puolen mukaisesti. Kallioon porattujen reikien on tietyllä varmuudella kestettävä ympäristön ja olosuhteiden muutokset rikkoutumatta, jotta ydinjätekapselit eivät vaurioituisi niiden loppusijoitusaikana.

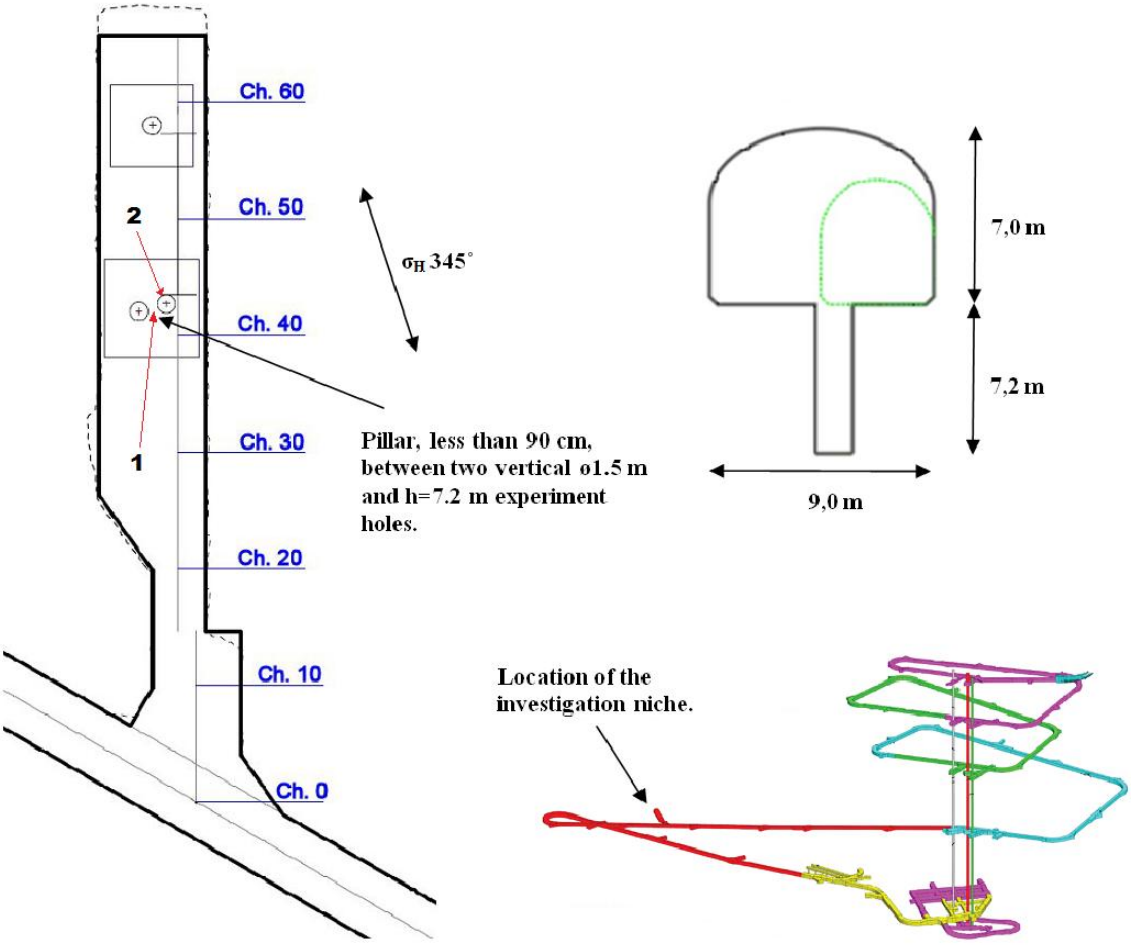


Kuva 2.2. *Loppusijoitustunneli ja loppusijoituskapseli, lähde: Posiva Oy ja Posiva Oy/Jussi Partanen.*

Kallion luontainen jännitystila (*in situ* jännitystila) muuttuu kun kallioon louhitaan tunneleita tai porataan erilaisia reikiä. Louhitun tilan ympärillä *in situ* jännitys kiertää louhitun tilan ja samalla se keskittyy tietyille kohdille. Lisäksi ydinjätekapselien loppusijoituksen alkuvaiheessa kapseleista siirtyy ympäristöön lämpöenergiaa, mikä vielä osaltaan lisää kiveen kohdistuvaa rasitusta. Nämä voimat puristavat kiveä, joka

ympäröi porattua reikää tai louhittua tunnelia. Mikäli jännitystila ylittää kiven hilseilylujuuden, kiveen syntyy jännitystilavaurio. Kameroiden avulla seurataan sitä, missä vaiheessa kivi alkaa vaurioitua kun jännitystilan voimakkuutta kasvatetaan tutkimuksen ensimmäisessä vaiheessa poraamalla viereinen reikä sekä mahdollisen jatkotutkimuksen aikana ympäröivää kiveä lämmittämällä.

Tutkimuksen aikana porattiin kolme tutkimusreikää (halkaisija 1,524 m ja syvyys 7,2 m), joista kaksi liittyivät diplomityössä käsiteltävään kamerajärjestelmään. Kuvassa 2.3 nuolella 2 merkitty reikä porattiin aiemmin ja reikään asennettiin mittauslaitteet. Mittalaitteiden asennuksen jälkeen aiemmin poratun reiän viereen porattiin uusi reikä, jolloin kallioperän paineen muuttuminen kuvassa 2.3 nuolella 1 merkityssä kohdassa aiheuttaisi kiven hilseilemisen.



Kuva 2.3. *POSE-tutkimustilan pohjakuva ja porattujen tutkimusreikien sijainti. Kamerat sijaitsevat nuolella 2 merkityssä reiässä. Lähde: Posiva Oy.*

Tutkimuksen kannalta on oleellista, että nähdään missä vaiheessa hilseilyä tapahtuu, jotta muiden instrumenttien tulosten perusteella voidaan määrittää, minkä suuruisilla voimilla hilseilyä alkaa tapahtua. Tutkimuksessa on myös varauduttu jatkotutkimuksiin, siinä tapauksessa että kiviaines ei lähde hilseilemään pelkän porauksen vaikutuksesta. Jatkotutkimuksen aikana kallioperää lämmitetään porattujen reikien viereen kairattuihin

reikiin asennetuilla lämmittimillä, jolloin lämpölaajeneminen kohottaa jännitystä ja kallio saadaan rikkoutumaan.

2.3 Työn aihe

Työn lähtökohtana oli rakentaa kameralaitteisto, joka kuvaa kiven muodonmuutoksia POSE-tutkimuksen porausten ja jatkotutkimusten aikana. Kuvamateriaalin keräämisen jälkeen diplomityöhön kuului aineiston käsittelyyn sopivan ohjelmiston kehittäminen ja aineiston käsittely helposti tulkittavaan muotoon. Kameroiden avulla pyritään näkemään, kuinka kivi käyttäytyy jännityksen muuttuessa. Kamerajärjestelmän on oltava riittävän tarkka, mutta kuitenkin tutkittavalta alueelta suurehkolta alalta kuvaava. Hyvällä resoluutiolla kuvatusta materiaalista nähdään helpommin, milloin kivessä alkaa tapahtua muutoksia, ennen kuin muutokset ovat radikaaleja.

Projektin alkuvaiheessa oli tarkoituksena selvittää soveltuvan kamerajärjestelmän saatavuus ja koota järjestelmä, joka vastaa mahdollisimman hyvin järjestelmälle asetettuja vaatimuksia hinnan kuitenkin pysyessä tutkimuksen kannalta kohtuullisena. Kamerajärjestelmän kasauksen jälkeen järjestelmän ohjaukseen toteutettiin ohjelmisto, joka toteutti tutkimukselle sopivan aikavälikuvauksen usealla kameralla.

Projektin toisessa vaiheessa kuvien käsittelyä varten tehtiin ohjelma, joka käsitteli kuvat ja muodosti tuloksista erilaisia videotiedostoja. Ohjelma oikaisee sylinterin pinnalta otetut perspektiivi-vääristyneet kuvat tasokuviksi, yhdistää useamman kameran kuvat yhtenäiseksi kuva-alueeksi ja muodostaa näistä kuvista halutulta kohdalta ja aikaväliltä videotiedoston.

2.4 Kuvauskohde

Kuvauskohde sijaitsee POSE-tutkimustilassa. Kuvattava kohde on 7,2 m syvä ja 1,524 m halkaisijaltaan oleva porattu sylinterinmuotoinen reikä. Kuvassa 2.4 näkyy kuvaan 2.3 nuolella 2 merkitty kuvattava kohde, kamerajärjestelmä kuvaa reiän vasemmanpuoleista seinustaa ja kamerajärjestelmä ja valot ovat asennettuna oikeanpuoleisessa trussissa. Kuvassa näkyy poratun reiän pohjalle kertynyttä kosteutta, joka lisää omat haasteensa kuvauskohteeseen, lisäksi kuvauskohteessa oleva pöly on omiaan sotkemaan kuvaamista.

Lähtökohtaisesti olosuhteet luolassa eivät ole kovinkaan suotuisat kameralaitteille ja elektroniikalle. Lisäksi viereisen reiän poraamisessa käytettävä vesi täytyi eristää mittalaitteita sisältävästä reiästä. Mittausten ajaksi kuvauskohde oli suljettuna kannella, jossa olivat läpiviennit kaapeloinneille. Reikään saattaa päätyä vettä myös kalliossa olevien rakojen kautta ja lattiapinnan tasoituksessa käytetyn betonin ja kallion välisen sauman läpi. Pohjalle mahdollisesti kertyvää vettä varten reiän pohjalle oli asetettu pumppu ja poistoputki.

Kuva 2.4. *Kuvauskohde, lähde: Posiva Oy/Kimmo Kemppainen.*

Reiän poraamiseen käytetty laite on melko massiivinen, johtuen porattavasta materiaalista ja reiän suuresta koosta. Kuvassa 2.5 on kuva reiän poraamiseen käytetystä koneesta POSE-tutkimustilassa.

Kuva 2.5. *Tutkimusreikien poraamiseen käytetty kone, lähde: Posiva Oy*

Kuvattava kohde oli valaistu hyvälaatuisilla loisteputkilla. Tutkimuskohteesta ei ollut datakaapelointeja maanpinnalle ja kuvauslaitteiston järjestelmän osien tuli olla akkuvarmennettuja sähkökatkosten varalta. Sähkölaitteiden osalta oli varauduttava matalaan 10 – 15 asteen lämpötilaan ja mahdollisiin kosteuden aiheuttamiin ongelmiin. Kaapeloinnin tuli yltää 20 – 40 metrin etäisyydellä olevaan mittauslaitteille varattuun konttiin.

2.5 Kameralaitteisto

Kameralaitteiston kuva-alan täytyi kattaa kappaleessa 2.4 esitellyn, poratun sylinterinmuotoisen reiän kehältä katsottuna noin 120 asteen levyinen, kuuden metrin korkuinen kaistale. Kuvausta varten tuli valita sopiva optiikka ja kamerat, joiden resoluutio ja kuva-ala vastaisivat vaatimuksia. Kuvien tallennus toteutettaisiin PC-ohjelmistolla, joka pystyy tallentamaan kuvat ja niiden kuvausajankohdat.

Järjestelmän tulisi pystyä tallentamaan kuvia hyvinkin pitkällä aikavälillä, jotta nähdään muutokset useamman viikon aikana tai mahdollisesti kuukausien aikavälillä. Jotta kuvia saadaan tallennettua näin pitkällä aikavälillä, järjestelmän on oltava riittävän luotettava ja kuvamateriaalille on oltava riittävästi tallennuskapasiteettia.

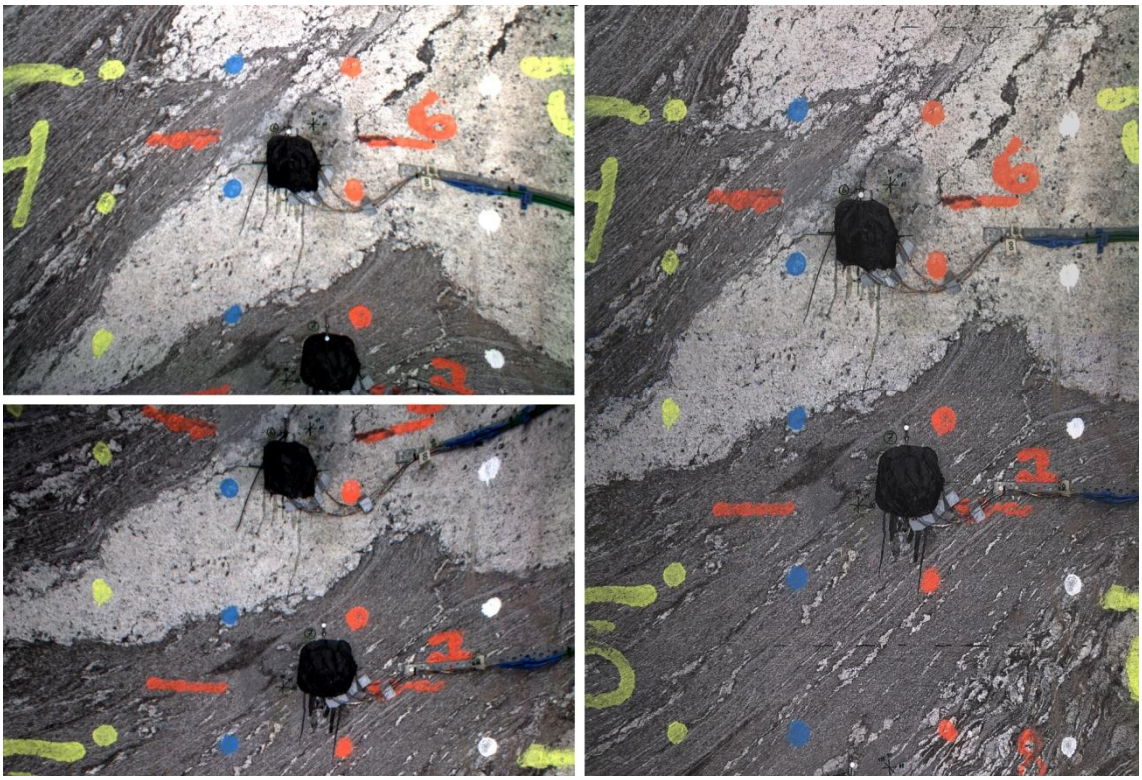
Kuvamateriaalin tallennuksen jälkeen eri kameroiden kuvat tulisi yhdistää yhtenäiseksi kuva-alueeksi ja näistä yhdistelmäkuvista laadittaisiin videotiedostoja, joista voitaisiin nähdä muutokset kiven rakenteessa videomuotoisena aineistona. Lisäksi projektin edetessä kuva-aineistolle oli tarkoitus tehdä kuva-analyysia.

2.6 Kuvamateriaali

Alkuperäisessä määritelmässä kuvien laadulle annettiin seuraavanlaiset kriteerit. Kuvien resoluution täytyisi olla sellainen, että kuvissa näkyisivät noin yhden millimetrin levyiset raot. Kameroiden yhtenäisen kuva-alan tulisi olla pystysuunnassa 6000 mm ja vaakasuunnassa 120 astetta 1524 mm leveän sylinterin kehältä. Kuvien välillä tulisi olla 10 – 20 senttimetriä päällekkäisyyttä, jotta kuvien yhdistäminen onnistuu yhdeksi isoksi kuvaksi.

Kuvamateriaalia oli alkuperäisen suunnitelman mukaan tallennettava 2 – 3 viikon aikaväliltä kattaen yksittäisen mittaustapahtuman keston. Lisäksi järjestelmällä pitäisi olla mahdollista tallentaa kuvasarjoja myös myöhempien jatkotutkimusten yhteydessä. Kameroiden kuvaus toimisi automaattisesti 5 – 60 sekunnin välein, riippuen käytettäväksi kuvaväliksi valitun aikajakson pituudesta.

Kuvamateriaalin keräyksen jälkeen eri kameroilla otetut kuvat oikaistaan ja yhdistetään kuvaksi, joka kattaa koko kuvattavan kohteen alueen. Peräkkäisistä kuvista voidaan muodostaa videotiedostoja halutuista kohdista ja aikaväleiltä. Kuvien yhdistämisen ja videotiedostojen muodostamisen lisäksi kuvia käsitellään niin, että ne ovat mahdollisimman hyviä kuvanlaadultaan ja selkeydeltään.



Kuva 2.6. *Vasemmalla kahden eri kameran tallentamien kuvien ala- ja yläosa. Oikealla puolella on kahdesta kuvasta yhdistetty kuva.*

Kuvassa 2.6 on nähtävillä ero käsittelemättömien ja käsiteltyjen kuvien välillä. Vasemmalla olevista kuvista voi nähdä, kuinka eri kuvien välillä on perspektiivistä johtuva vääristymä, koska kuva on otettu sylinterinmuotoisesta pinnasta. Oikealla puolella on lopputulokseksi saatu kuvien yhdistelmä, kahden kameran kuvien välillä.

2.7 Aikataulu

Projekti käynnistyi helmikuussa 2010 ja kamerajärjestelmän tuli olla käytettävissä kesäkuussa 2010, jolloin mittauksien ensimmäisen vaiheen oli tarkoitus alkaa. Kuvien käsittelyyn tarkoitetun ohjelmiston luomiseen, kuvien käsittelyyn ja tämän diplomityön kirjoittamiseen oli varattu aikaa vuoden 2010 loppuun asti.

Mittauksen kesto, jolloin järjestelmän tuli toimia oli noin 2-3 viikkoa. Järjestelmän tuli myös toimia myöhempien jatkotutkimusten yhteydessä. Lopulta järjestelmä päätettiin toteuttaa joustavasti niin että kuvasarjoja voi kuvata useita tai kuvataajuutta voi muuttaa kesken mittauksen.

3 MENETELMÄT

Luvussa esitellään varsinaisessa työssä käsiteltyjä menetelmiä ja tapoja, joilla dataa on käsitelty ja tallennettu. Yleisesti käytössä olevat algoritmit on esitelty pääpiirteittäin ja ohjelmaan itse kehitetyt algoritmit on käyty läpi tarkemmin. Osa työssä käytetyistä menetelmistä on jätetty käsittelemättä niiden yleisluontoisuuden vuoksi, tai koska ne ovat tietokonegrafiikassa yleisesti käytössä ja niiden toteutukset ovat löydettävissä useista lähteistä.

3.1 HD Photo

HD Photo on Microsoftin kehittämä kuvanpakkausformaatti, joka tukee häviöllistä ja häviötöntä pakkaustapaa. Formaatin nimi on vaihtunut useamman kerran ja se tunnetaan nykyisin nimellä JPEG XR, aiemmin nimenä on ollut Windows Media Photo. Tässä dokumentissa käytetään HD Photo nimeä, koska ohjelmien käyttämä pakkauskirjasto käyttää HD Photo määritelmän mukaista pakkaustapaa.

HD Photo käyttää tehokasta häviötöntä kaksisuuntaista väriavaruusmuunnosta, kaksisuuntaista päällekkäistä biortogonaalista muunnosta ja kehittynyttä ei-aritmeettista entropiakoodausta. Näiden tekniikoiden yhdistelmä tarjoaa erittäin hyvän pakkaustehokkuuden niin että häviöt kuvan sisällössä jäävät pieniksi. Käytännössä formaatti on melko monimutkainen toteutukseltaan, mikä on osaltaan saattanut vaikuttaa siihen, ettei vapaassa levityksessä olevia koodekkeja ole tarjolla. [4]

HD Photon ominaisuuksia ovat häviötön ja häviöllinen pakkaus ja hyvä pakkaussuhde. Lisäksi formaatti tukee erilaisia värimalleja sekä kuvapikselin bittimääriä. Kehittyneempiä ominaisuuksia ovat kuvien paloittainen käsittely sekä erilaiset operaatiot kuten pakkaussuhteen muutos ja kuvan kääntö ilman kuvan täydellistä purkamista ja uudelleenpakkaamista.

3.2 MPEG-4 videoformaatti

MPEG-4 ei ole itsessään pakkausformaatti, vaan se on enemmänkin kokoelma erilaisia määritelmiä liittyen kuvan ja äänen pakkaukseen. MPEG-4 standardi määrittelee suuren määrän erilaisia multimedian pakkaukseen ja hallintaan liittyviä osa-alueita. MPEG-4 formaatin sisältämän laajan pakkausformaattien ja muiden ominaisuuksien valikoiman vuoksi standardiin on määritelty erilaisia profiileja, joiden perusteella voidaan valita sovellukselle tai laitteelle sopiva osa standardista, jolloin koko standardin mukaista järjestelmää ei tarvitse toteuttaa.

Toteutetussa ohjelmassa MPEG-4 formaatista on valittu käyttöön MPEG-4 part 2 ASP-profiili, (Advanced Simple Profile), joka määrittelee käytetyn pakkausjärjestelmän. FFmpeg ohjelmakirjastosta käytetty Xvid koodekki noudattaa tämän profiilin mukaista pakkaustapaa. MPEG-4 video voidaan tallentaa eri

säiliötiedostoformaateissa, tässä yhteydessä video tallennetaan AVI-tiedostoformaatin määrittelemässä säiliötiedostotyypissä. MPEG-4 formaatti on kohtalaisen laaja ja esimerkiksi MPEG-4 part 2 mukainen pakkausformaatti on eri kuin MPEG-4 part 10 määrittelemä pakkausformaatti. Tässä työssä käytetään MPEG-4 part 2:n mukaista pakkaustapaa kokonaiselta nimeltään MPEG-4 part 2 Advanced Simple Profile.

ASP profiilin mukainen pakkaustapa käyttää MPEG-1:n ja MPEG-2:n tavoin diskreettiä kosinimuunnosta lohkoittaiseen pakkaukseen. Lisäksi videota pakataan laskemalla kokonaisten kuvakehysten väliltä vain muutoksia sisältävät osat. Formaatti tukee edellisen kehyksen perusteella laskettavaa muutoslaskentaa (P-frames) sekä kaksisuuntaista menneen ja tulevan kokokuvan (I-frame) välistä muutoslaskentaa (B-frames). Lisäksi formaatti tukee lohkojen liikkeentunnistuksesta, joka parantaa pakkaussuhdetta, sekä kokonaisen kuvan liiketunnistuksesta.

3.3 Gammakorjaus

Gammakorjauksessa muutetaan kuvien värien kirkkausarvoja epälineaarisesti. Gammakorjausta voidaan käyttää useampaankin eri tarkoitukseen, mutta tässä yhteydessä korjausta käytetään korostamaan joko kirkkaita tai vaaleita kohtia kuvissa. Gammakorjauksen avulla voidaan tuoda tummien alueiden sisältämä kuvainformaatio silmille havaittavaksi tai vastaavasti kirkkaiden alueiden sisältävä informaatio voidaan tummentaa helpommin havaittavaksi. Kuvassa 3.1 näkyy tavallisen lineaarisen kirkkauskorjauksen ja gammakorjauksen ero. Gammakorjauksessa yläpään kirkkausarvot eivät juuri muutu ja pysyvät näkyvissä, verrattuna kirkkaus-kontrasti korjaukseen, jossa alun perin kirkkaat väriarvot näkyvät suurilla tasomuutoksilla tasapaksuina vaaleina alueina.

Kuva 3.1. *Gammakorjausesimerkki. Vasemmalla alkuperäinen kuva, keskellä kirkkaus-kontrasti säädetty kuva ja oikealla gammakorjattu kuva.*

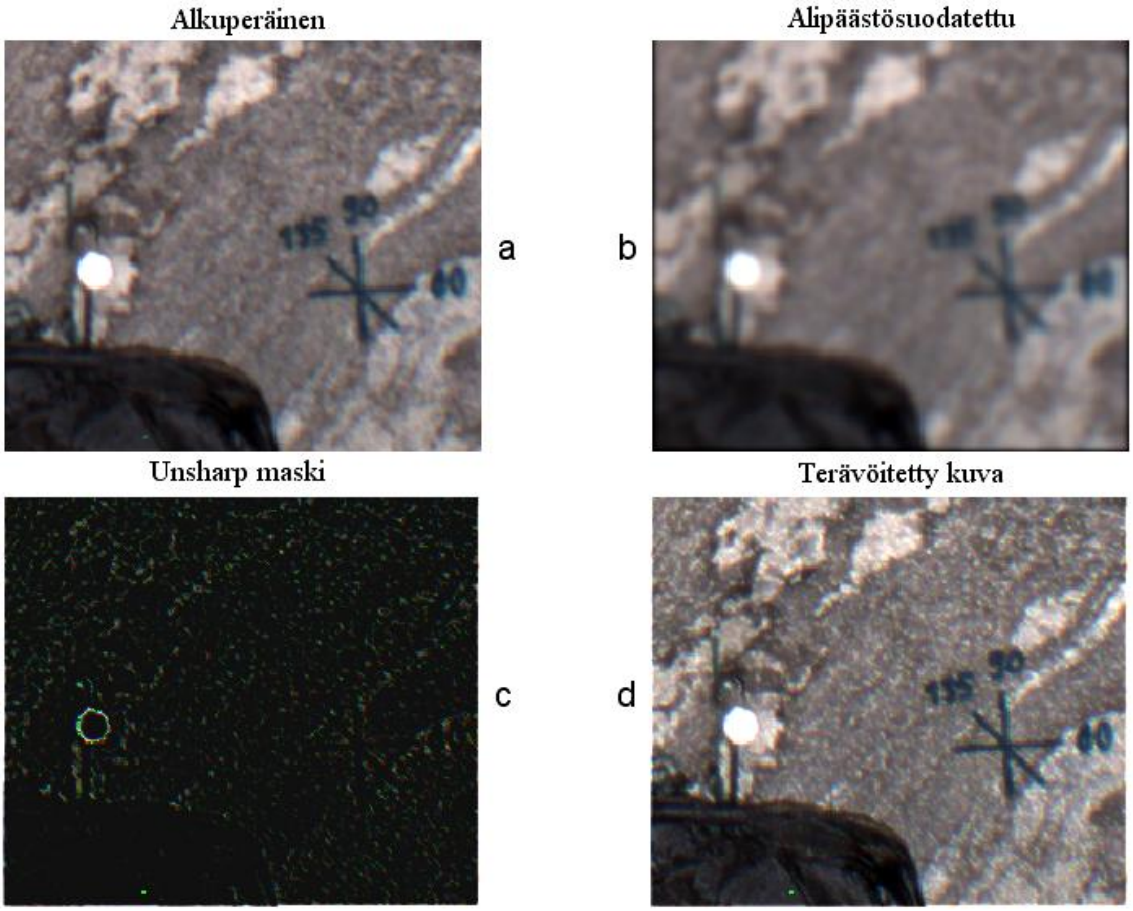
Korjauksen toiminta perustuu siihen, ettei silmä pysty erottamaan kaikkia kuvan sisältämiä kirkkausarvoja, jotka ovat lähellä toisiaan. 8-bittisen RGB kuvan dynamiikka on huomattavasti laajempi, kuin mitä silmällä voidaan havaita, kun kuvaa tarkastellaan esimerkiksi monitorilta. Gammakorjauksessa muutetaan kirkkausarvojen dynamiikkaa, jolloin esimerkiksi tummalla alueella olevien kirkkausarvojen erot ovat huomattavasti

suurempia kuin vaaleilla alueilla, jolloin silmä voi tulkita kirkkauserot tummilla alueilla toisistaan poikkeaviksi. Gammakorjaus muuttaa arvoja ainoastaan alueella, jossa arvoja halutaan tuoda esille ja jättää kuvassa olleet kirkkaat / tummat arvot lähes muuttumattomiksi. Gammakorjauksen toteuttava yhtälö on esitetty kappaleessa 3.7 valoisuuskorjauksen yhteydessä.

3.4 Unsharp maskaus

Unsharp-masking tekniikassa kuvaa terävöitetään summaamalla alkuperäiseen kuvaan alkuperäisestä kuvasta erotettu vain korkeataajuuksiset komponentit sisältävä kuva. Alkuperäiseen kuvaan lisättävä korkeat taajuudet sisältävä kuva luodaan alkuperäisestä kuvasta alipäästösuodattamalla ja vähentämällä alipäästösuodatettu kuva alkuperäisestä, jolloin saadaan tarvittava terävöitysmaski.

Varsinainen tarkennus tällä tekniikalla saadaan aikaiseksi lisäämällä jollain halutulla terävöityskertoimella kerrottu maskauskuva alkuperäiseen kuvaan. Operaation jälkeen kuva näyttää alkuperäistä kuvaa tarkemmalta, koska korkeataajuuksiset komponentit, eli terävät muutokset, ovat kuvassa alkuperäistä suuremmilla poikkeamilla. Matalataajuuksiset eli laajat samankaltaiset kuva-alueet taas pysyvät lähes muuttumattomina.



Kuva 3.2. *Unsharp maskaus*

Kuvassa 3.2 unsharp maski c lasketaan alan kirjallisuudessa [6; 7] esitetyllä algoritmilla

$$c(x,y) = a(x,y) - b(x,y) \qquad (1)$$

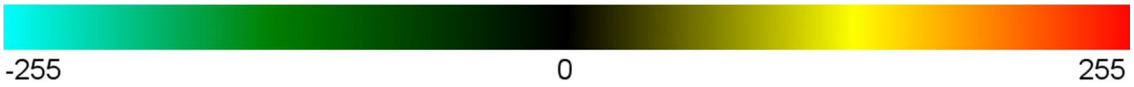
ja lopputulos

$$d(x,y) = a(x,y) + c(x,y) \cdot r \qquad (2)$$

jossa r on terävöitysmaskin voimakkuuskerroin, a on alkuperäinen kuva ja b on ylipäästösuodatettu kuva. Sumennettu kuva b voidaan laskea eri tavoilla. Esimerkissä sumennetun kuva tekoon on käytetty gaussiaanista suodatinta.

3.5 Erotuskuvien laskenta

Erotuskuvien avulla voidaan havainnollistaa, kuinka kiven pinnassa tapahtuu muutoksia kahden eri aikana otetun kuvan välillä. Erotuskuvat lasketaan valitsemalla kuvien aikajanalta jokin referenssikuva, jonka suhteen tarkasteltavan aikajakson muiden kuvien erotus lasketaan. Erotuskuvalle saadut uudet väriarvot kuvataan takaisin RGB-väriavaruuteen värikartan avulla. Väriarvojen kuvaamiseen käytetty värikartta näkyy kuvassa 3.3.



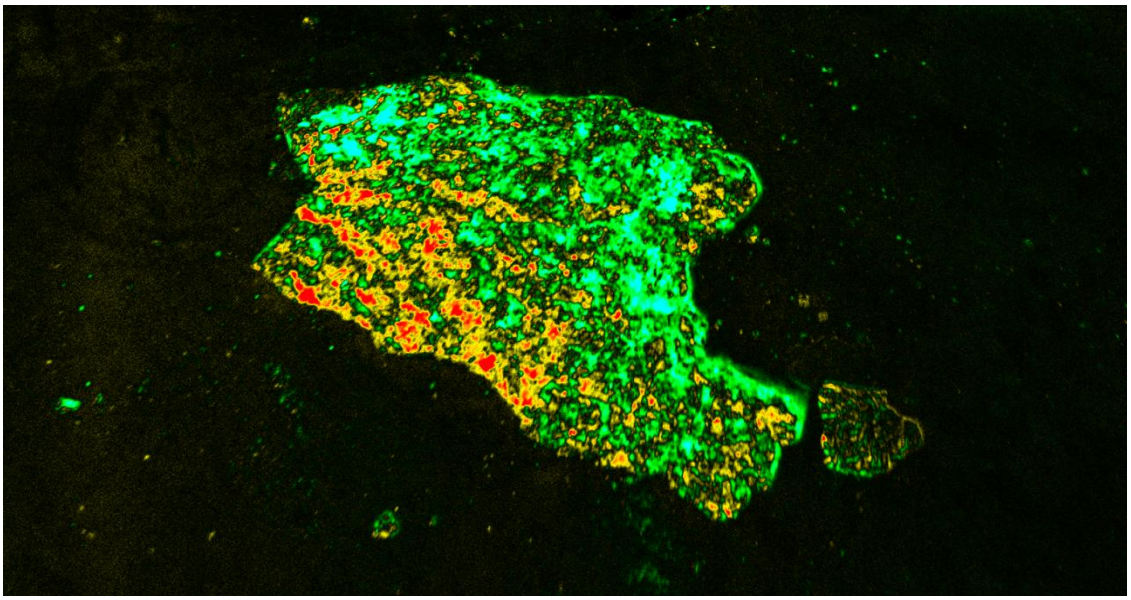
Kuva 3.3. *Erotuskuvan värikartta.*

Kahden kuvan, joiden kuvapisteiden väriarvot ovat positiivisia lukuja välillä $[0{,}255]$, välinen erotus palauttaa arvo-alueen, joka on kaksinkertainen alkuperäiseen nähden. Tässä tapauksessa alkuperäinen arvoalue on $[0, 255]$ ja erotuskuvan arvoalue $[-255, 255]$. Saatu tulos muutetaan negatiivisella puolella liukuvasti vihreästä syaanin väriseksi ja positiivisella puolella liukuvasti keltaisesta punaiseksi kuvan 3.3 värikartan mukaisesti. Erotuskuva lasketaan kaavalla

$$c(x,y) = r \cdot \sum_{k=1}^{3} [a_k(x,y) - b_k(x,y)] \qquad (3)$$

jossa a ja b ovat kaksi eri ajanhetkellä otettua kuvaa ja r on vahvistuskerroin. Muuttuja k indeksoi väritasoja.

Kuvassa 3.4 näkyy erotuskuvassa kallioseinämästä irti lyödyn testipalasen aiheuttama muutos kuvien välillä. Valonlähteen suunnasta riippuen irronneen kiviaineksen kohdalla kuvan väriarvot ovat muuttuneet joko pienemmiksi tai suuremmiksi. Näistä väriarvojen muutoksista muodostuu erotuskuvassa näkyvä värikartan avulla väritetty muutoskuva.



Kuva 3.4. *Erotuskuvan esimerkki.*

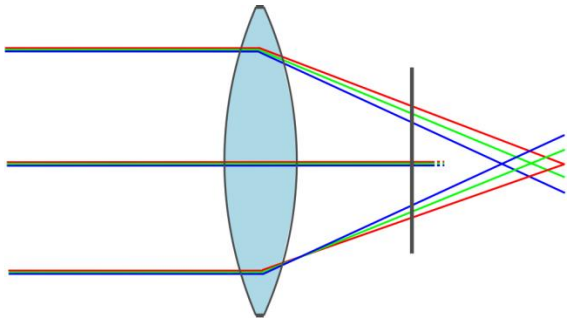
Erotuskuvan laskentaan on tässä toteutuksessa kaksi vaihtoehtoa: RGB värikomponenttien erotuksen keskiarvo tai RGB värikomponenttien erotuksen summa. Kiviaineksen ollessa pääsääntöisesti harmaata näillä kahdella vaihtoehdolla ei ole juurikaan eroa ja keskiarvon tapauksessa voidaan muuttaa pelkästään vahvistuskerrointa pienemmäksi, jolloin saadaan sama vaikutus kuin laskemalla keskiarvo. Erotuksen laskennan jälkeen on mahdollista säätää vahvistuskerrointa, jolloin pienemmätkin muutokset saadaan näkyviin.

3.6 Kromaattisen virheen korjaus

Kuviin aiheutuu sivuttainen kromaattinen virhe muodostuu, koska eriväriset (tai paremminkin eri taajuuksiset) säteet kulkevat omaa reittiänsä linssin läpi, vaikka niillä olisi sama lähtöpiste. Linssin taittokerroin n on riippuvainen aallonpituudesta ja tällöin linssin polttovälin f on myös oltava riippuvainen aallonpituudesta [5]. Ohuen linssin yhtälöstä

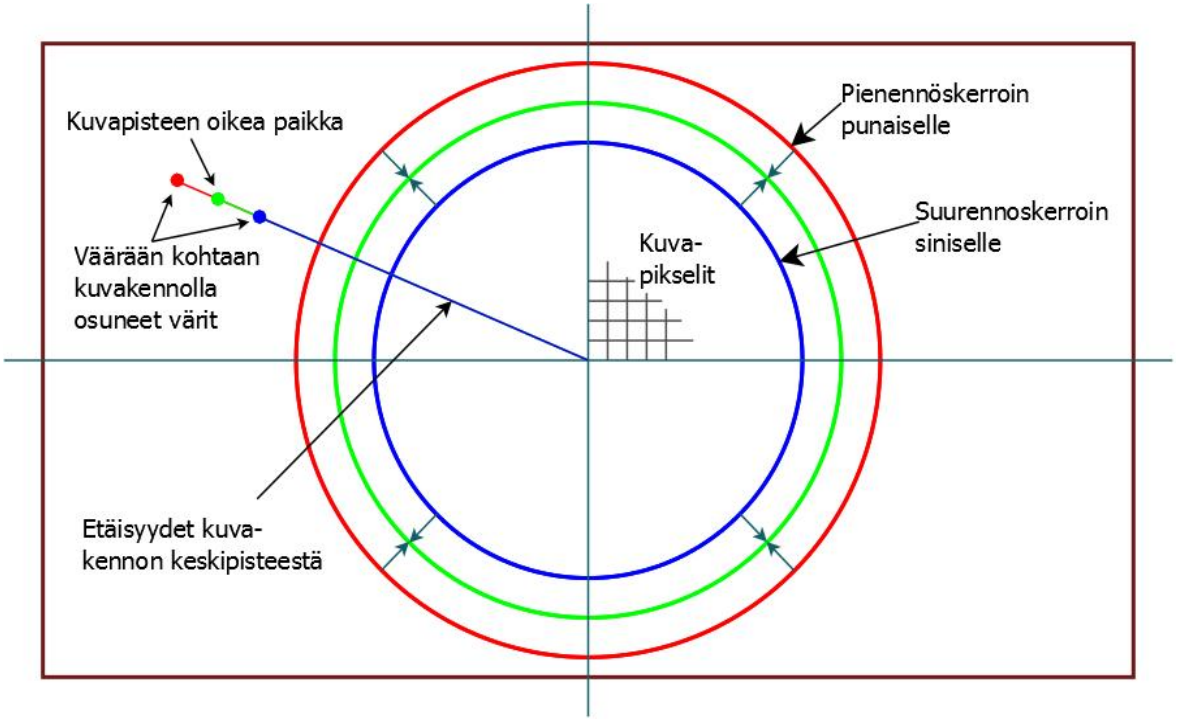
$$\frac{1}{f} \approx (n-1)\left(\frac{1}{R_1} - \frac{1}{R_2}\right) \qquad (4)$$

nähdään että linssin taittokerroin n vaikuttaa linssin puoliskojen kaarevuussäteiden R_1 ja R_2 lisäksi siihen, kuinka paljon valo taittuu kulkiessaan linssin lävitse. Kuvassa 3.5 samasta pisteestä lähteneet säteet ovat kulkeneet eri reittiä eri taittokertoimen vuoksi ja ne päätyvät kameran kennolla keskipisteestä katsoen eri etäisyyksille.



Kuva 3.5. *Aallonpituudesta riippuvien taittokertoimien vaikutus valonsäteisiin.*

Kromaattinen virhe aiheuttaa kuvan reunoilla olevien kohteiden reunoihin värivirheitä. Virhe korostuu varsinkin lyhyen polttovälin linsseillä. Virhe voidaan korjata zoomaamalla punaista ja sinistä pikselitasoa. Korjauksen jälkeen samasta pisteestä kuvattavaa kohdetta lähteneet valonsäteet ovat kuvan pikselitasolla samassa kohtaa, jolloin kuvan resoluutio saadaan hieman paremmaksi.



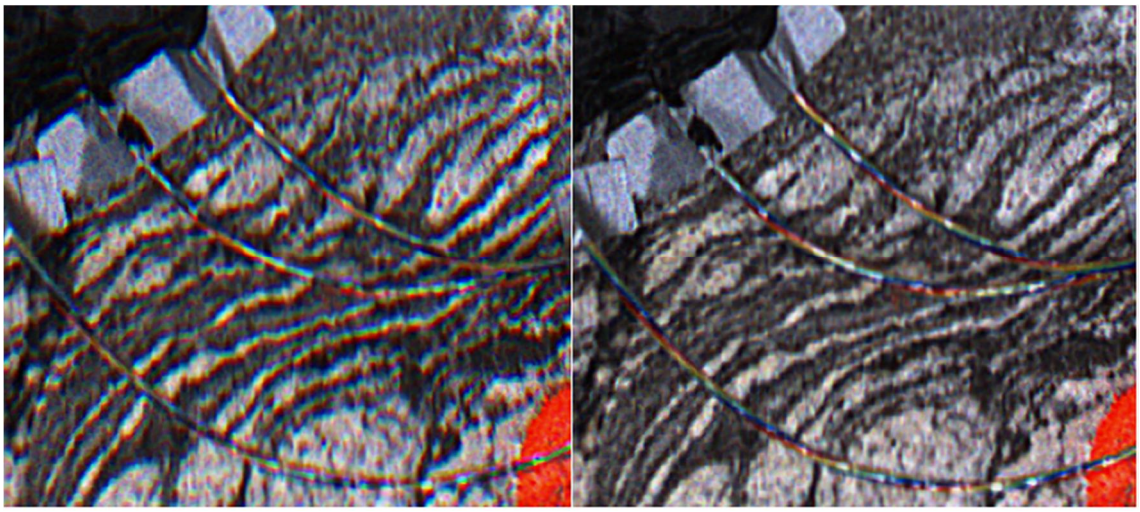
Kuva 3.6. *Kromaattinen virhe kuvakennolla katsottuna.*

Virheen korjaamiseksi käytetään erisuuruista pisteen paikkakerrointa sinisellä ja punaisella värikanavalla. Punainen väritaso saa pienennyskertoimen ja sininen väritaso suurennuskertoimen, vihreän väritason kertoimen pysyessä vakiona. Suurennuskerroin ilmoitetaan kuvan keskipisteestä reunojen suuntaan, koska linssin keskipisteessä kaikilla tasoilla on yhtenäinen piste, jossa taittokerroin on sama. Kuvassa 3.6 keskellä on piste, jossa kaikkien kennolle saapuneiden säteiden paikka on sama. Jos kennolla liikutaan keskipisteestä pois, huomataan että samasta pisteestä lähteneet eri värit alkavat saada eri paikan kennon väripikseleissä, vaikka niiden pitäisi sijaita samassa kohtaa.

Varsinainen korjaus suoritetaan käyttämällä erilaisia projektioita tekstuurien värikomponenteille kuvien oikaisun ja yhdistämisen aikana. Käytännössä käyttämällä

eri projektiota eri väritasoille saadaan sama vaikutus kuin väritasojen suurentamisella keskipisteen suhteen. Kuvassa 3.6 operaatio olisi sama kuin sinisen kehän suurentaminen ja punaisen kehän pienentäminen.

Paras paikka korjauksen suorittamiselle on sensorin muodostaman RAW-kuvan interpolointivaihe, jolloin kuvakennon yhdessä tasossa oleva eri värien kuvapistekuvio muodostetaan tasavälisiksi kuvatasoiksi. Tässä vaiheessa kuvapisteisiin kohdistetaan interpolointioperaatioita, jolloin syntyy vääristymä, jota ei voida korjata enää myöhemmin täydellisesti. Käytännössä jouduttiin tyytymään korjaamaan jo interpoloituja kuvia, koska RAW-kuvien tallentaminen olisi vienyt liian paljon tilaa.



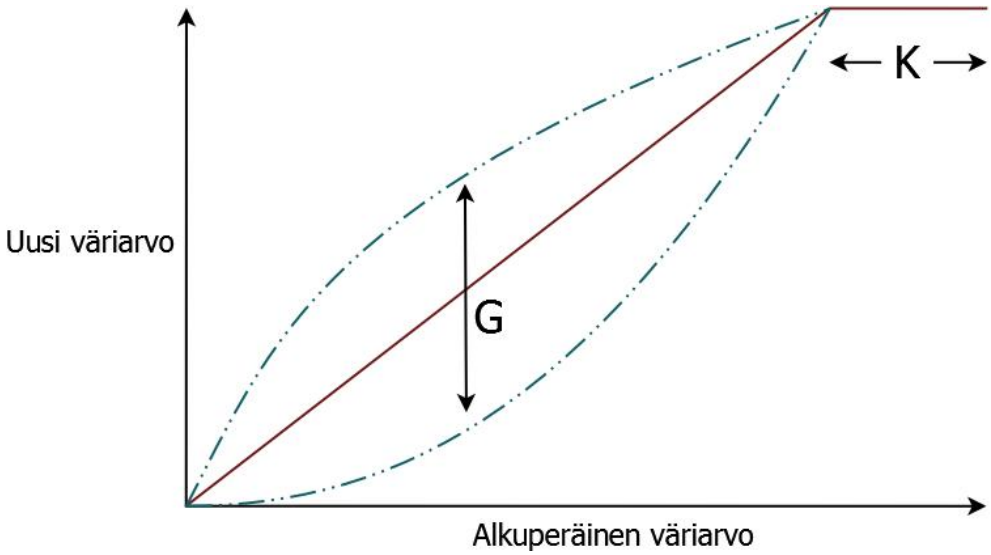
Kuva 3.7. *Vasemmalla olevassa lähtökuvassa on kromaattinen virhe näkyvissä ja oikealla puolella on korjattu kuva.*

Kuvassa 3.7 näkyy vasemmallaa puolella kromaattisen virheen aiheuttama punainen ja sininen reuna terävien muutosten ympärillä. Virhe näkyy silmälle ainoastaan reunojen ympärillä, koska kohdissa, joissa värimuutokset ovat pieniä tai alueet ovat tasavärisiä, ei väritasojen värikomponenttien siirtyminen näy silmämääräisesti. Kumpikin puoli kuvasta on jälkikäsitelty kirkkauden, kontrastin ja värikylläisyyden osalta samalla tavalla havainnollisuuden parantamiseksi. Kromaattisen virheen korjaukseen käytetty tekniikka esitellään tarkemmin luvussa 3.10.

Pitkittäistä kromaattista virhettä ei tässä yhteydessä korjattu, koska sen vaikutusta ei havaittu kuvissa ja sen korjaaminen on vaikeampaa, koska informaatiota on menetetty enemmän virheen luonteen vuoksi. Pitkittäisessä kromaattisessa virheessä linssin muodostamat kuvat eri värialueilla fokusoituvat eri polttoväleille, jolloin jos esimerkiksi vihreä kuvataso fokusoidaan kennolla, punainen ja sininen väritaso päätyvät pois hyvästä fokusoinnista. Huonosta fokusoinnista johtuen ei-fokusoituneiden tasojen optinen resoluutio huononee ja on käytännössä mahdotonta palauttaa digitaalisesti. Kuvien terävyyttä parannettiin jo muiden kuvankäsittelyoperaatioiden yhteydessä, joten osa korjattavissa olevasta pitkittäisestä kromaattisesta virheestä tulee korjattua tässä yhteydessä.

3.7 Valoisuuskorjaukset

Kuvien kirkkaus ja kontrasti säädetään jokaiselle kuvalle erikseen, jotta saadaan mahdollisimman tasainen yhdistetty kuva. Kameroiden linsseissä on hieman rakenteellista eroa ja toisaalta esimerkiksi linssin aukon kokoa on mahdotonta säätää käsin täysin samaksi, jolloin kuvien kirkkauksiin tulee poikkeamaa. Lisäksi valaistus ei ole kuvien välillä täysin yhtenevää.



Kuva 3.8. *Väriarvot uusiin pisteisiin kuvaava säätökäyrä.*

Kuvan kirkkaus säädetään epälineaarisesti gamma- ja kirkkaussäätöjen yhdistelmällä. Kuvan kirkkaus säädetään kuvassa 3.8 kirkkauskäyrän säätöparametrin K avulla. Kirkkaus korjataan ennen gammakorjausta, jotta gammakorjauksen säätöparametri saadaan toimimaan tässä yhteydessä halutulla tavalla, yllä olevan kuvan mukaisesti parametrilla G. Lopuksi on vielä mahdollista säätää erikseen koko kuvan kontrastia. Valoisuuskorjaus lasketaan kaavalla

$$c(x,y) = \left(\frac{a(x,y)\cdot K}{R}\right)^{\frac{1}{G}} \cdot R \quad (5)$$

jossa parametri K vastaa kirkkauden säätöä, vakio R on kuvapisteen värikomponentin bittimäärän muodostaman arvoalueen maksimi ja parametri G gammakorjauskerroin.

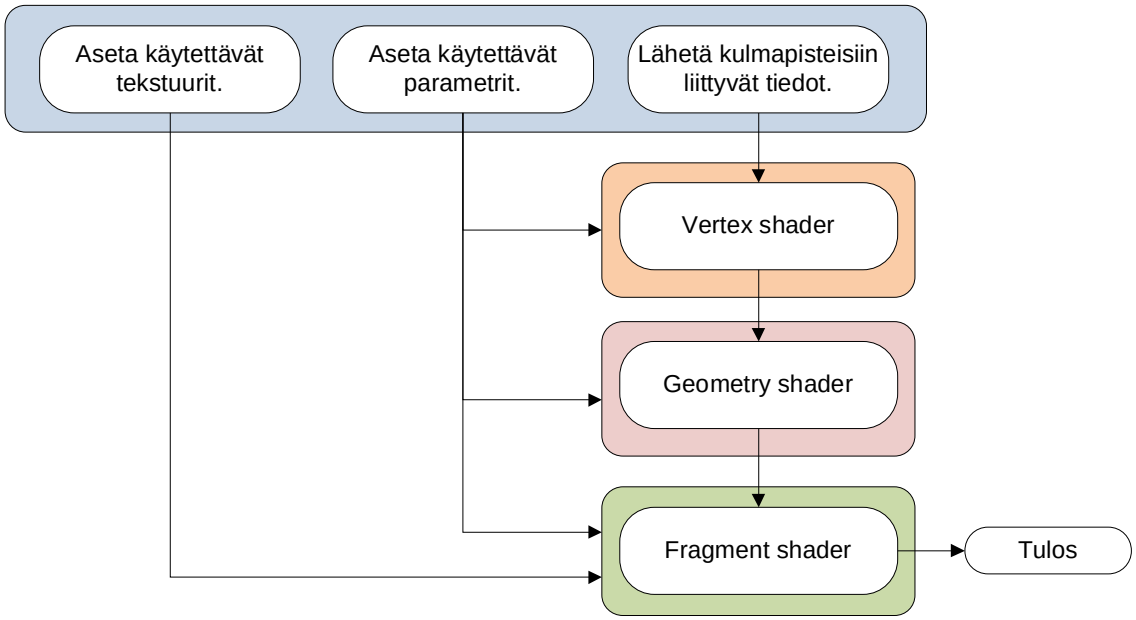
Valoisuuskorjaukset toteutettiin RGB tasossa, vaikka normaalisti kuvan kirkkautta säädetään esimerkiksi HSV-väriavaruudesa, jossa muutetaan pelkästään kuvapisteen intensiteettikomponenttia, värikomponenttien pysyessä samana. Koska kuvadata oli lähestulkoon harmaata ja kromaattinen virhe haluttiin korjata myöhemmin, kohdistettiin kirkkausmuutokset RGB tasoihin toisistaan riippumatta, jolloin väritasojen välille ei muodostu riippuvuutta, joka sotkisi kromaattisen virheen korjauksen.

3.8 OpenGL laskenta

OpenGL on laajalti tunnettu ja yleisesti käytetty vektorigrafiikkakirjasto. Rajapinnan kautta päästään käsiksi laitteistokiihdytettyyn laskentaan, jos järjestelmässä on käytettävissä OpenGL yhteensopiva näytönohjain. Ohjelmassa käytettyjä ominaisuuksia tukevat käytännössä kaikki vuoden 2008 jälkeen valmistetut näytönohjaimet.

Ohjelmassa käytettiin OpenGL-rajapinnan määrittelyn versiota 3.0. Uudemmissa OpenGL versioissa piirto-operaatioihin käytetään GLSL (OpenGL Shading Language) ohjelmointikieltä näytönohjaimen prosessoinnin ohjaamiseen. GLSL on myöhemmin lisätty OpenGL-grafiikkakirjastoon näytönohjaimien kehittymisen myötä, tarjoamaan käyttäjille monipuolisemmat laskentamahdollisuudet ja suuremman kontrollin grafiikan laskentaprosessiin. GLSL-ohjelmat jakaantuvat minimissään kahteen osaan, joista pakolliset ovat vertex shader ja fragment shader ohjelmat. Kolmas ohjelma on myöhemmin GLSL-määrittelyyn lisätty geometry shader, jota ei käytetty tässä yhteydessä.

Vertex shader ohjelmassa käsitellään kulmapisteiden (vertex) yhteyteen liitetyt arvot ja parametrit, jotka välitetään seuraavaan shader-ohjelmaan käsiteltyinä. Ohjelmassa kulmapisteille voidaan tehdä normaalisti esimerkiksi kierto-, siirto- ja projektio-operaatioita. Muille ohjelmaan saapuneille tiedoille ja parametreille voidaan suorittaa halutunlaisia operaatioita ja tulokset välitetään eteenpäin käsiteltäviksi. Kuvassa 3.9 esitellään pääpiirteittäin nykyisen OpenGL-grafiikkakirjaston piirtoprosessi käyttäjän näkökulmasta.



Kuva 3.9. *OpenGL ja GLSL 3.3 laskenta.*

Ennen fragment shader ohjelmaa näytönohjain yhdistää vertex shaderin tiedot kolmioiksi ja interpoloi tiedot, jotta fragment shader ohjelma voidaan suorittaa. Fragment shader ohjelma käsittelee vertex shaderilta saapuvista interpoloiduista

kuvapistettä koskevista tiedoista ja muista parametreista saaduilla tiedoilla jokaisen kuvapikselin, joka päätyy lopputuloksessa kuva- tai muuksi dataksi.

3.9 Virtuaalikamera

Virtuaalisella kameralla tarkoitetaan tässä yhteydessä kameraa, joka kuvaa numeerisesti esitettyä kolmiulotteista maailmaa. Tavallisessa reaalimaailmassa toimivassa kamerassa on optiikan ja fyysisten rakenteiden aiheuttamia rajoitteita, joita ei pystytä kiertämään teknisillä ratkaisuilla, ainakaan järkevillä raha- tai työmäärillä. Numeerisesti määritellyssä kolmiulotteisessa maailmassa toimivalla virtuaalisella kameralla ei tällaisia rajoitteita ole.

Virtuaalisen kameran avulla saadaan kuvattua kolmiulotteisessa maailmassa oleva kappale hyvin erilaisilla tavoilla kuin olisi mahdollista normaalilla kameralla. Virtuaaliselle kameralle voidaan asettaa erilaisia projektioita, joista käyttökelpoisimmat ovat perspektiivi ja ortogonaaliprojektio. Perspektiiviprojektiolla saadaan aikaiseksi samanlainen kuva kuin saataisiin fyysisellä kameralla ja linssillä. Perspektiiviprojektiossa etäämpänä olevat kohteet näkyvät pienempänä tuloksessa. Ortogonaaliprojektiolla puolestaan saadaan aikaiseksi kuva, joka litistää kuvattavan kohteen niin että kameran edessä olevan kohtisuoran tason normaalin suuntaiset syvyyssuuntaiset etäisyydet kuvautuvat samaan tasoon. Toisin sanoen ortogonaaliprojektiossa ei ole syvyysvaikutelmaa, jolloin kuvattavassa suunnassa olevat kohteet ovat samankokoisia vaikka ne olisivat toisistaan kauempana.

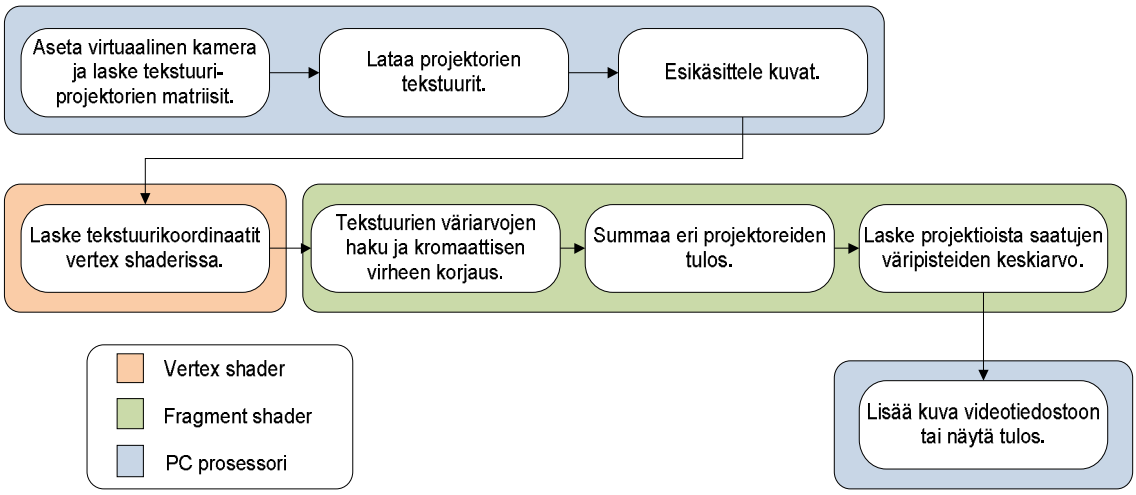
3.10 Kuvien oikaisu

Kuvien oikaisu tapahtuu kolmiulotteisessa avaruudessa laskettavilla projektioilla. Alkuperäiset kameroiden paikat asetetaan tekstuuriprojektorien paikoiksi ja uudeksi virtuaalisen kameran pisteeksi asetetaan jokin haluttu mielivaltainen piste avaruudessa. Virtuaalisen kameran paikasta kuvataan mallia, joka vastaa reaalimaailmassa kuvatun kohteen muotoa. Tämän mallin pintaan projisoidaan tekstuuriprojektoreilla alkuperäisten kameroiden ottamat kuvat, jolloin tulokseksi saadaan reaalimaailmassa olevan sylinterin pintaa vastaava kuvio, olettaen että parametrit on asetettu oikein.

Aluksi laskenta toteutettiin käyttämällä *ray tracing* algoritmia, jossa virtuaalisesta kamerasta lähetettyä sädettä seurattiin kuvatusta kohteesta muodostetun 3D mallin kautta alkuperäisen kameran projektiopisteeseen. Nopeampaa toteutustapaa etsittäessä kävi ilmi, että saman operaation saa toteutettua tavallisella 3D piirtoalgoritmilla, jota käytetään esimerkiksi näytönohjaimissa.

Näytönohjaimen käyttöön perustuvassa algoritmissa tekstuurien projisointi toteutetaan mallin kulmapisteiden muuntamisella tekstuurikoordinaateiksi projektion avulla. Seuraavassa vaiheessa näytönohjain yhdistää kulmapisteet kolmioiksi, jonka jälkeen voidaan laskea yksittäisiä pisteitä fragment shaderissa kolmion pinnalla. Kulmapisteistä saadut tekstuurikoordinaatit interpoloidaan näytönohjaimen toimesta

ennen fragment shaderia, jolloin saadaan yksittäisten kolmioiden pinnalla olevien kuvapisteiden koordinaatit tekstuuriprojektorin tekstuurilla. Kuvassa 3.10 on kuvien oikaisuun käytetyn algoritmin rakenne vuokaaviona. Kaaviossa on eroteltu väreillä, missä ohjelman osassa algoritmin osa toimii ja mitä kyseisessä algoritmin kohdassa lasketaan.



Kuva 3.10. *Kuvien käsittelyyn, oikaisuun ja yhdistämiseen käytetyn algoritmin vuokaavio.*

Kromaattisen virheen korjaus vertex ja fragment shaderin avulla tapahtuu käyttämällä jokaiselle värikomponentille omaa tekstuuriprojektoria. Vertex shaderiin syötetään parametreina R, G ja B komponenteilla hieman eri FOV:lla olevat tekstuuriprojektorit, jolloin tekstuurikoordinaattien laskennassa väritasot saadaan takaisin päällekkäin. Fragment shaderissa eri värikanaville käytetään eri tekstuurikoordinaatteja, jolloin saadaan haettua R, G ja B värikomponentit oikeista paikoista ja kromaattinen virhe saadaan korjattua.

Koodiesimerkissä 3.1 Vertex shader koodissa lasketaan tekstuuriprojektiot ja muunnetaan kulmapiste virtuaalisen kameran näkymään. Tekstuurikoordinaatit lasketaan jokaiselle tekstuuriprojektorille kulmapisteestä projektorin edessä olevalle tasolle projektion avulla. Tulos tallennetaan *texcoord* jonoon, joka välittyy fragment shader ohjelmaan. Virtuaalikameran kuvassa näkyvän kolmion kulmapiste lasketaan virtuaalikameran näkymään kameran kohdistusmatriisin *mvm* ja projektiomatriisin *pm* avulla. Kulmapiste siirretään laskentaketjussa seuraavaan vaiheeseen *gl_Position* muuttujan kautta.

```
#version 330
in vec4 vertex;
uniform mat4 mvm; //Kameran kohdistusmatriisi
uniform mat4 pm; //Projektiomatriisi
uniform mat4 projectors[15]; //5 * 3 tekstuuriprojektoria
out vec4 texcoords[15]; //Laskettavat tekstu. koordinaatit

void main(void)
{
   for( int i=0; i<15; i++)
   {
      vec4 tmp = vec4( vertex.xyz, 1.0);
      //Laske tekstuuriprojektio kulmapisteestä
      //kolmion pinnalle
      texcoords[i] = projectors[i] * tmp;
   }
   //Laske kulmapisteen projektio virtuaalikameraan.
   gl_Position = pm * mvm * vertex;
};
```

Koodiesimerkki 3.1. *Vertex-shader ohjelma.*

Koodiesimerkin 3.2 fragment shader ohjelma käsittelee yhtä virtuaalikameran näkymän kuvapistettä kerrallaan, kuvapistettä vastaavan kolmion pinnalla. Fragment shader koodissa haetaan kolmion pinnalta interpoloitujen tietojen perusteella kolmion pinnalle kuviointi tekstuuriprojektoreista. Tekstuurien pintakuvio haetaan `texture2DProj` funktiolla, joka laskee nelikomponenttisesta tekstuurikoordinaatista litistetyn kaksiulotteisen koordinaatin ja palauttaa kaksiulotteisesta tekstuurista koordinaattia vastaavan kuvapisteen väriarvon. Tekstuuriprojektoreiden tiedot yhdistetään *blending*-kohdassa. Kromaattinen virhe korjautuu tekstuurien värikomponentteja haettaessa, koska jokaiselle värille käytetään tässä tapauksessa omaa tekstuuriprojektoria, eri FOV:lla.

```
#version 330
uniform sampler2D texture0, texture1, texture2, texture3,
  texture4;
out vec4 outpix;
in vec4 texcoords[15];

void main(void)
{
  outpix = vec4( 0.0, 0.0, 0.0, 1.0 );
  vec4 tex[5];
  //Poista projektorin takaprojektio tarkistamalla suunta
  if( texcoords[0].q > 0.0 )
  {
    //Hae pikselin väriarvo komponenteittain, chrom korj.
    tex[0].r = texture2DProj(texture0, texcoords[0] ).r;
    tex[0].g = texture2DProj(texture0, texcoords[1] ).g;
    tex[0].b = texture2DProj(texture0, texcoords[2] ).b;
  }
  else
    tex[0] = vec4( 0.0, 0.0, 0.0, 1.0 );

//Muutama samankaltainen tekstuurihaku poistettu.
//Tekstuureja ei voi indeksoida nykyisen glsl:n silmukoissa.
```

```
  if( texcoords[12].q > 0.0 )
  {
    tex[4].r = texture2DProj(texture4, texcoords[12] ).r;
    tex[4].g = texture2DProj(texture4, texcoords[13] ).g;
    tex[4].b = texture2DProj(texture4, texcoords[14] ).b;
  }
  else
    tex[4] = vec4( 0.0, 0.0, 0.0, 1.0 );

  //BLENDING
  vec4 divs = vec4( 0.0, 0.0, 0.0, 1.0 );
  //Yhdistä kuvat painotetun keskiarvon mukaisesti
  for( int i=0; i<5; i++ )
  {
    //Pisteen etäisyys teks. keskipisteestä max normilla
    float dist = max( abs( texcoords[i*3].x /
         texcoords[i*3+1].w -0.5),abs( texcoords[i*3+1].y/
         texcoords[i*3+1].w - 0.5 ) );

    #define BFACTOR 0.5
    //Blending painokerroin
    float weight=1-pow(dist,BFACTOR)/pow(0.5,BFACTOR)

    if( tex[i].r > 0.0 ) {
       tex[i].r *= weight;
       divs.r += weight;
    }

    if( tex[i].g > 0.0 ) {
       tex[i].g *= weight;
       divs.g += weight;
    }
    if( tex[i].b > 0.0 ) {
       tex[i].b *= weight;
       divs.b += weight;
   }
    outpix += tex[i];
 }
    outpix /= divs;
}
```

Koodiesimerkki 3.2. *Fragment-shader ohjelma.*

Vertex- ja fragmentshader ohjelmat käännetään suoritettavaksi koodiksi OpenGL-rajapinnan käskyillä. Ohjelmat käännetään näytönohjaimen toimesta vasta juuri ennen käyttöä, jolloin niistä tulee kyseisellä näytönohjaimella toimiva koodi, joka on optimoitu mahdollisimman nopeaksi kyseisessä laitteistossa. Shader ohjelmien kääntämisen tuloksena saadaan yksi ohjelma joka sisältää kaikki laskentaan tarvittavat käskyt. Ohjelma voidaan valita käytettäväksi ohjelmaksi ennen haluttua piirto-operaatiota. Tämä mahdollistaa että voidaan käyttää useita eri shader-ohjelmia samassa ohjelmassa.

3.11 Kuvien yhdistäminen

Kuvien yhdistäminen toteutetaan näytönohjaimen avulla kuvien oikaisun yhteydessä. Kameroiden kuvat kohdistetaan sijoittamalla tekstuuriprojektorit paikoilleen kolmiulotteiseen avaruuteen mahdollisimman tarkasti. Projektoreiden ollessa oikeissa paikoissaan ja jos niiden projektiosuunta on oikein, kuvat ovat päällekkäisiltä osiltaan kohdallaan. Kuvien päällekkäin menevät osat yhdistetään samaksi kuvaksi laskemalla päällekkäisten kuvapikseleiden keskiarvo.

Käytännössä kuvat eivät ole täysin yhtenevät päällekkäisiltä osiltaan, koska ne kuvaavat epätasaista pintaa ja pinnalla olevia kappaleita eri suunnilta. Tästä johtuen kuvat täytyy yhdistää laskemalla päällekkäisille osille jokin sopiva sekoitusfunktio, joka tasaa erot kahden kuvan välillä. Sekoitusfunktio kehitettiin sellaisen idean ympärille, jossa painotetulla keskiarvolla lasketaan kuvapisteiden arvo kahden kuvan väliltä. Painokertoimeksi laskettiin tekstuurikoordinaattien keskipisteestä etäisyys kuvapisteeseen maksiminormin mukaan ja laskettu etäisyys muutettiin vielä käänteisesti laskevaksi käyräksi.:

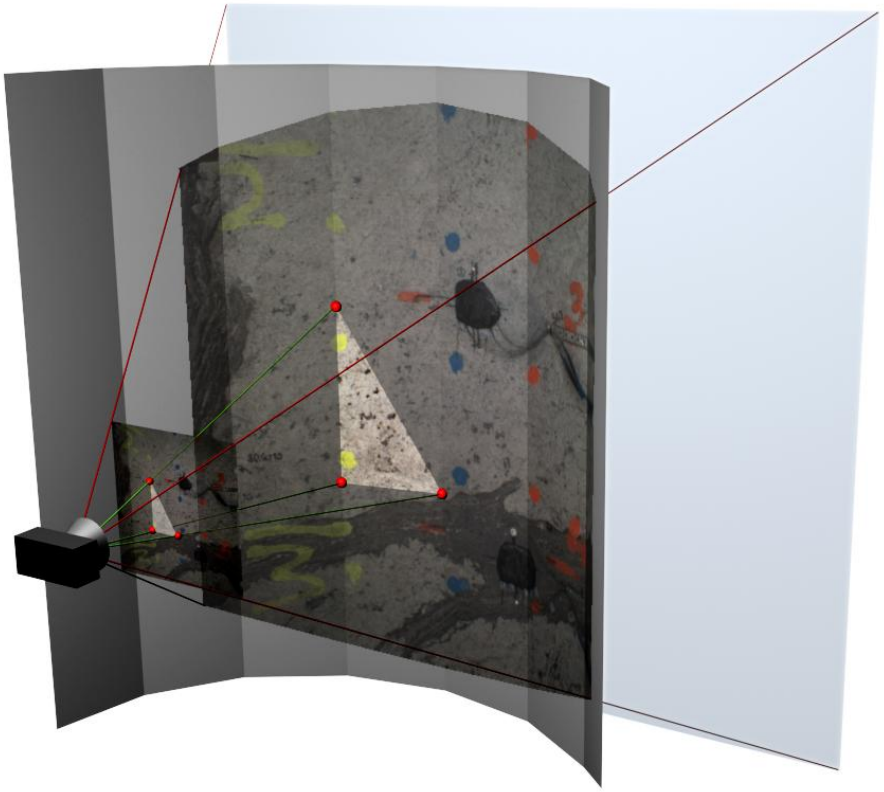
$$p = 1 - |\max(\bar{x} - 0.5)|^r - 0.5^r \qquad (6)$$

Kaavasta saadaan tekstuurikoordinaatille laskettua painokerroin p, joka vastaa kuvapisteessä käytettävän tekstuurin epävarmuustekijää. Vektori $\bar{x}$ sisältää kaksiulotteisen tekstuurikoordinaatin. Parametri r tekee käyrästä sopivan jyrkän, r:n ollessa < 1. Jos toisen tekstuurin kuvapisteellä on suurempi painokertoimen arvo kuin toisella, ensimmäisen tekstuurin kuvapistettä summautuu enemmän lopputulokseen. Maksiminormin käyttö euklidisen sijaan takaa painokertoimen pysymisen välillä $[0,1]$ ja mahdollistaa kuvien yhdistämisen kaikilta sivuilta.

Fragment shader koodiesimerkissä 3.2 *blending*-kohdassa $\bar{x}$ lasketaan perspektiivijaolla tekstuurikoordinaateista, jolloin perspektiivinen tekstuurikoordinaatisto muutetaan kaksiulotteiseksi tasokoordinaatistoksi. Painotetulla keskiarvolla voidaan yhdistää $0 - n$ kappaletta tekstuureja. Tässä yhteydessä kuitenkin päällekkäin tulee olemaan vain kaksi tekstuuria kerrallaan. Jos yhtään tekstuuria ei ole päällekkäin, muodostuu väriksi musta, jonka väriarvo on 0 kaikkien komponenttien osalta. Väri tulee tekstuurin reunavärin määrittelystä OpenGL tekstuurien tilassa.

3.12 Tekstuuriprojektorit

Tekstuuriprojektorien matriisit laskettiin lähteessä [8] sivulla 5 esitellyllä objektiavaruuden tekstuurikoordinaattilaskennalla. Lähteessä esitetyllä algoritmilla saadaan projisoitua objektiavaruuden kulmapisteet tekstuuriprojektorin avaruuteen. Nämä pisteet litistetään vielä perspektiiviprojektiolla tekstuuriprojektorin edessä olevalle tasolle, jolloin tulokseksi saadaan laskettavana olleet kaksiulotteiset tekstuurikoordinaatit. Kaksiulotteisia tekstuurikoordinaatteja voidaan lopulta käyttää kuvapikseleiden hakuun kaksiulotteisen tekstuurin pinnalta.



Kuva 3.11. *Tekstuuriprojektorin toiminta*

Kuvassa 3.11 näkyy tekstuuriprojektorin toimintaa havainnollistava esimerkki. Projektorin toiminta on käänteinen verrattuna siihen mitä varsinaisessa mielessä projektorista voisi olettaa. Objektiavaruudessa kappaleen pinnalla olevat pisteet, joita kuvassa merkitään punaisilla palloilla, siirtyvät perspektiiviprojektiossa projektorin edessä olevalla tasolle, jolta pintakuvio haetaan kohteen pinnalle.

Tekstuuriprojektori voidaan muodostaa seuraavan matriisioperaation avulla:

$$T_o = \begin{bmatrix} 1/2 & 0 & 0 & 1/2 \\ 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot P_p \cdot V_p \cdot M \qquad (7)$$

Tekstuuriprojektorimatriisi T_o on matriisi jolla kulmapisteet muunnetaan vertex shaderissa tekstuurikoordinaateiksi. P_p on projektorin projektiomatriisi ja V_p on projektorin paikan ja suunnan määrittävä näyttömatriisi. Numeerisena esitetty matriisi siirtää ja skaalaa koordinaatiston tekstuurin koordinaatistoon. Projektiomatriisin ja näyttömatriisin tulo $P_p \cdot V_p$ on projektorin kuvatasolle projisoitu piste. Jos piste on välillä $[-1,1]$, se tulee näkymään lopputuloksessa. Tekstuurin koordinaatistossa vastaava väli on $[0,1]$.

Kaavasta 7 voidaan tässä yhteydessä poistaa matriisi M. Mallin kääntö-siirto-skaalausmatriisia M ei tarvita, koska oikaisujen laskennassa käytetyt objektiavaruudessa olevat kappaleet ovat staattisia ja niiden lopulliset paikat on määritetty mallin tekovaiheessa. Matriisin M poistamisen jälkeen päädytään muotoon:

$$T_o = \begin{bmatrix} 1/2 & 0 & 0 & 1/2 \\ 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot P_p \cdot V_p \qquad (8)$$

Tekstuuriprojektorilla kerrottu kulmapiste voi saada suurempia arvoja, kuin väli $[-1,1]$, jolloin välin ulkopuolella olevat pisteet täytyy rajata pois. Pisteiden rajaus tapahtuu OpenGL:n avulla. OpenGL tekstuurin parametreissa määritellään, että tekstuurissa välin [0,1] ulkopuolelle menevät arvot asetetaan tiettyyn arvoon, tässä tapauksessa nollaksi, joka jätetään huomioimatta kuvien yhdistämisvaiheessa. Toinen huomioitava seikka on että perspektiiviprojektiomatriisi muuntaa lineaariavaruuden toiseksi avaruudeksi, jolloin kaikki projektorin taustapuolella olevat pisteet litistyvät projektorin kuvatasolle. Tämä estetään tekemällä tarkastus fragment shaderissa. Projektorin takana olevilla pisteillä on eri etumerkki, joka tarkastetaan ennen perspektiivijakoa, mikä litistää kolmiulotteisen koordinaatiston kaksiulotteiseksi. Tarkastuksessa negatiivisen arvon omaavat pisteet hylätään.

Vääristymien korjaamisen tarkkuus riippuu käytetyn mallin kolmioverkon tiheydestä. Kulmapisteiden määrittämän kolmion pinnalla korjausta ei lasketa, vaan tekstuurikoordinaatit lasketaan lineaarisesti interpoloimalla. Käytännössä tämä tarkoittaa sitä että on käytettävä riittävän tiheää kolmioista muodostuvaa mallia. Kun kolmiot ovat riittävän pieniä, ovat paikalliset virheet kolmioiden pinnalla merkityksettömän pieniä suhteessa koko kuvassa olevaan vääristymään.

Menetelmän huonoja puolia on, että alkuperäisten kameroiden paikat ja suunnat olisi hyvä tietää mahdollisimman tarkasti. Ilman minkäänlaista käsitystä kameroiden paikoista on hyvin työlästä alkaa kokeilemalla selvittämään, millä arvoilla kuvat saadaan päällekkäin. Tässä toteutuksessa oletettiin että kameroiden paikat tunnetaan, joten ohjelmaan ei luotu mitään automatiikkaa kohdistukseen. Ohjelmassa kameroiden parametreja voidaan säätää reaaliaikaisesti, jolloin nähdään heti muutosten vaikutus ja näin ollen kuvien kohdistaminen on hieman helpompaa.

4 JÄRJESTELMÄN TOTEUTUS

Tässä luvussa käsitellään järjestelmän toteutukseen liittyviä tietoja, eli miten, miksi ja millä tavalla on päädytty käyttämään kyseistä osaa tai menetelmää. Kappaleessa esitellään myös järjestelmän osaksi päätyneiden ratkaisujen rakenteet ja lopputulokset.

4.1 Kamerat ja linssit

Kameroiden tärkeimmäksi valintaperusteeksi muodostui kameran sulkimen kestävyys. Järjestelmän kameran tuli kestää yli sadantuhannen kuvan ottaminen, joka rajoitti tavalliset mekaanisella sulkimella varustetut kamerat pois niiden mekaanisen sulkimen toimintavarmuuden vuoksi. Mekaanisen sulkimen sijaan valittiin kamerat, joissa on elektroninen suljin. Valitussa hintaluokassa oli elektronisella sulkimella varustettuja kameroita vain pienikokoisella C tai CS – koon linssikiinnikkeellä. C ja CS – tyyppisissä kiinnikkeissä linsseissä on kierretyyppinen kiinnitys kameran runkoon ja niille sopivat kameran kuvakennon lävistäjän mitat ovat 1/2 ” ja 2/3 ”. Koska kuvakennot olivat linssityypin rajoituksen takia pieniä, niistä aiheutui ongelmia lyhyen polttovälin saavuttamiseksi.

Kuva 4.1. *Objektiivi ja kamera, kuvat valmistajien sivuilta.*

Kameratyypiksi valittiin kuvassa 4.1 esitetty 10 megapikselin IDS GmbH:n valmistama uEye UI-1490 SE, joka on varustettu USB 2.0 liitännällä. Kamerassa oli riittävä, 3840 x 2748 resoluutio, elektroninen suljin, sopiva hinta, sekä riittävän suuri kenno mahdollisimman hyvän FOV:n saavuttamiseksi. Lisäksi kamerat olivat erittäin pienikokoisia ja kamerat voitiin sijoittaa melko vapaasti. Kameran koko ilman linssiä oli 34 mm x 32 mm x 27,4 mm. Linssi kiinnitettynä kamera vei suurin piirtein kaksinkertaisen tilan.

Optiikan löytäminen kameroihin osoittautui hieman vaikeammaksi sovitussa hintaluokassa, jossa kennojen koot ja objektiivit rajoittuivat C-kiinnityksellisiin kameroihin. Tarvittavan kuva-alan saavuttamiseksi linssiltä vaadittiin mahdollisimman tarkka optinen resoluutio sekä lyhyt polttoväli. Järjestelmän hankinta-ajankohtana ei

ollut saatavana riittävän hyviä C-kiinnikkeen linssejä, joten hyvän resoluution saavuttamiseksi päädyttiin kompromissiin, jossa käytettiin hieman pienempää kuva-alaa ympyrän kehältä katsottuna.

Objektiiviksi valittiin käytettävän kennon kanssa 4,16 mm polttovälin saavuttava, kuvassa 4.1 näkyvä Lensation GmbH:n valmistama CMFA0420ND. Objektiivin optinen resoluutio ei ole aivan sopiva suhteessa kameran pikselien kokoon. Kameran pikselin koko on 1,67 µm ja objektiiville ilmoitettu optinen resoluutio 500 mm kuvausetäisyydellä on 7,4 µm. Optisen resoluution ollessa huonompi kuin kennon pistekoko kuvan kontrasti heikkenee, koska valonsäteet kohdistuvat usean kuvapisteen alueelle esimerkiksi linssin epätarkkuuksista tai diffraktiosta johtuen.

Käytännössä objektiivilla saavutettu erottelukyky pienemmällä aukon koolla ja kontrastin parannuksen/terävöityksen jälkeen pääsee lähelle kameran kennon resoluutiota. Linssin aukon koko säädettiin niin että linssin syvyysterävyys olisi niin suuri, että kohteessa eri etäisyydellä olevat kohdat ovat fokuksessa. Käytännössä syväterävyys oli melkein kaikilla aukon suuruuksilla hyvä, joten aukon koko valittiin niin että kuva oli mahdollisimman tarkka.

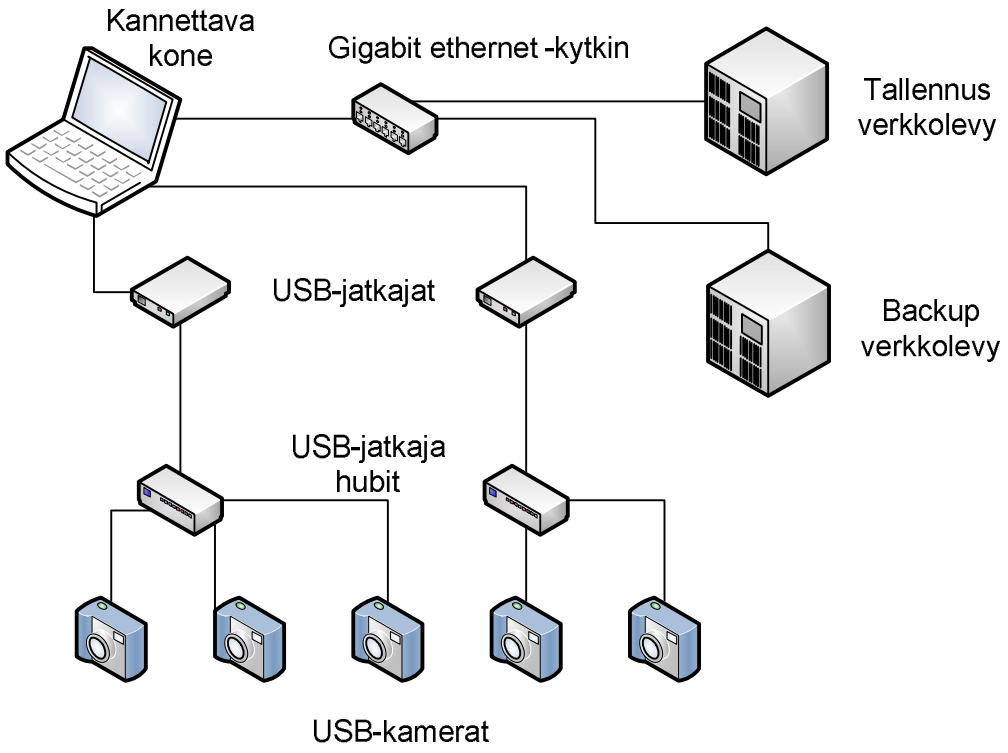
Linssin fokus asetettiin äärettömäksi, koska linssin työetäisyys oli 100 mm – ∞ ja linssin ääretön fokus alkaa noin 500 mm etäisyydeltä, kuvattavan kohteen ollessa yli 500 mm etäisyydellä. Linssin optinen resoluutio paranee hieman, kun fokus säädetään mahdollisimman kauas.

Linssin muodostama kuva on tarkin kun aukon koko on mahdollisimman pieni, kuitenkin niin että pieni aukon koko ei vielä muodosta diffraktiosta johtuvaa sumentumista. Aukon koko haettiin sopivaksi käsin tarkastelemalla sopivalla etäisyydellä olevaa testikappaletta, joka oli samanmuotoinen kuin kuvattava kohde. Linssejä testattaessa osoittautui, että linssien tarkkuus vaihteli, luultavasti johtuen valmistustoleransseista.

4.2 Kamerajärjestelmä

Kamerajärjestelmä koostuu kuvassa 4.2 näkyvillä olevista komponenteista. Laitteiden tilaushetkellä ei ollut vielä varmuutta, kuinka lyhyttä tallennusväliä järjestelmän tulisi tukea, joten kamerat päätettiin kytkeä USB-liitännän kautta ja kuvien tallennusjärjestelmä puolestaan Gigabit Ethernet verkon yli, jolloin liitäntäporttien tiedonsiirtonopeus ei muodostuisi pullonkaulaksi. Kaikki kuvassa 4.2 näkyvät kamerajärjestelmän osat on akkuvarmennettu UPS-laitteella.

Kamerajärjestelmän kamerat liitettiin kannettavan koneen USB-portteihin USB-hub-jatkajien kautta. USB Ranger 2104 USB-jatkajat pidentävät standardin määrittelemän viiden metrin maksimietäisyyden 100 metriin asti, jolloin USB-kamerat saadaan kytkettyä järjestelmään kuvauskohteessa, kameroita käyttävän kannettavan koneen sijaitessa mittalaitekontissa.



Kuva 4.2. *Kamerajärjestelmän rakenne*

Kuvien tallentamiseen käytetyt verkkolevyasemat kytkettiin kannettavaan koneeseen Gigabit Ethernet kytkimen kautta. Gigabit Ethernet valittiin käyttöön sen tarjoaman nopeuden vuoksi. Viiden kameran kuvista muodostuu pakattuna noin 65 MT kuvadataa ja kun samaan aikaan tallennetaan kuvien varmuuskopio, siirrettävää tietoa kertyy noin 130 MT. Käytännössä kuvien tallennus koneen keskusmuistista levylle kesti n. 2,5 sekuntia, johtuen levyasemien n. 30 MT/s maksimikirjoitusnopeudesta.

4.3 Valaistus

Mahdollisimman hyvän kuvanlaadun, sekä videotoiston saavuttamiseksi tarvittiin valaisimet, joiden valon spektri on mahdollisimman tasainen ja kattaa koko näkyvän valon spektrin alueen. Elektronisella sulkimella varustettujen kameroiden kuvassa näkyi tavallisten valaisimien välkyntä häiritsevästi, jolloin välkyntä olisi näkynyt lopputuloksen videotoistossa haitallisena aaltoiluna. Tämä ongelma poistettiin käyttämällä välkkymättömiä valaisimia.

Aluksi valaisimiksi kokeiltiin LED-loisteputkia, mutta niiden väritoisto ei ollut halutunlainen, jolloin päädyttiin käyttämään värintoistoindeksiltään hyviä Osramin Lumilux De Luxe -loisteputkia elektronisella sytyttimellä. Elektroninen sytytin poistaa loisteputkesta välkynnän käyttämällä korkeampaa vaihtovirran taajuutta. Korkeampi taajuus siirtää välkynnästä aiheutuvan aaltoilun niin korkeille taajuuksille, ettei kameroissa käytetyillä valotusajoilla aaltoilua enää esiinny.

4.4 Kameroiden ripustus, kaapelointi ja sähkönsyöttö

Kuvassa 4.3 järjestelmän kamerat, valaisimet ja USB-hubit kaapeleineen on kiinnitetty trussiin ja ovat valmiina siirrettäväksi kuvauskohteeseen. Järjestelmä koottiin valmiiksi ja nostettiin kuvauskohteeseen toimintakuntoisena, jolloin järjestelmän asennus oli huomattavasti helpompaa.

Kuva 4.3. *Kamerajärjestelmän kamerat ja valaisimet kasattuna ennen laitteiden siirtoa kuvauspaikkaan. Kuva Posiva Oy/Kimmo Kemppainen.*

Kamerat ja valaisimet kiinnitettiin trussiin, joka kiinnitettiin alapäästä kuvauskohteessa poratun sylinterin pohjaan ja yläpäästä sylinterin seinään. Trussia käytettiin myös kaikkien johdotusten ja USB-jatkajien kiinnityspaikkana. Kamerat kiinnitettiin trussiin sitä varten tehtyihin liikuteltaviin kiinnikkeisiin ja kameroiden kohdistusta helpotettiin kiinnittämällä kamerat kuulapää-kiinnikkeisiin.

Kaapelointi toteutettiin mahdollisimman pienellä kaapelimäärällä. Kamerat kytkettiin PC-koneeseen USB-hub-jatkajien kautta. USB-jatkaja muuntaa datasignaalit Cat-5 kierrettyyn pariin, jolloin USB:lle saadaan pidempi kantavuus ja parempi häiriönsieto. Datakaapeloinnin lisäksi kohteeseen vedettiin sähköt UPS:lta ja lisäksi asennettiin web-kamera valvontaa varten. Valaisimet kytkettiin niiden mukana olevilla pitkillä kaapeleilla suoraan pinnalla oleviin pistokkeisiin.

PC- ja muiden laitteiden sähkönsyöttö toteutettiin UPS-laitteen kautta. UPS-laite korjaa verkkovirran häiriöt ja muuntaa aaltomuodon oikeanlaiseksi. Lisäksi UPS-laitteessa on akku, joka pystyy syöttämään virtaa sen taakse kytkettyyn kuormaan mahdollisen sähkökatkon aikana. ONKALO:ssa esiintyy erilaisia sähköhäiriöitä ja katkoksia huoltotöiden ja paljon virtaa kuluttavien laitteiden johdosta. Valaistusta ei kytketty UPS:n taakse sen lisäämän suurehkon kuorman vuoksi.

4.5 Kuvien tallennusjärjestelmä

Kuvat tallennettiin kahdelle Western Digital ShareSpace neljän Teratavun NAS -levylle, joista toinen levy toimi varmuuskopiona. Alkuperäisestä määrittelystä poiketen kuvaväliksi valittiin minimissään 30s. Näin ollen PC-koneena riitti tavallinen kannettava kone. Viiden kameran ja 10 megapikselin kuvakoolla koneelta kesti kuvien hakemiseen, pakkaamiseen ja tallentamiseen noin 24 sekuntia, joka oli riittävä nopeus tässä tapauksessa.

Kuvien tallennusmuodoksi valittiin häviötön Microsoftin HD Photo -formaatti, jolloin kuvat saatiin tallennettua RGB- muodossa häviöttömänä suoraan kameralta. Tällä haettiin mahdollisimman pientä kohinan tasoa, jolloin erotuskuvissa olisi mahdollisimman vähän ylimääräistä kohinaa.

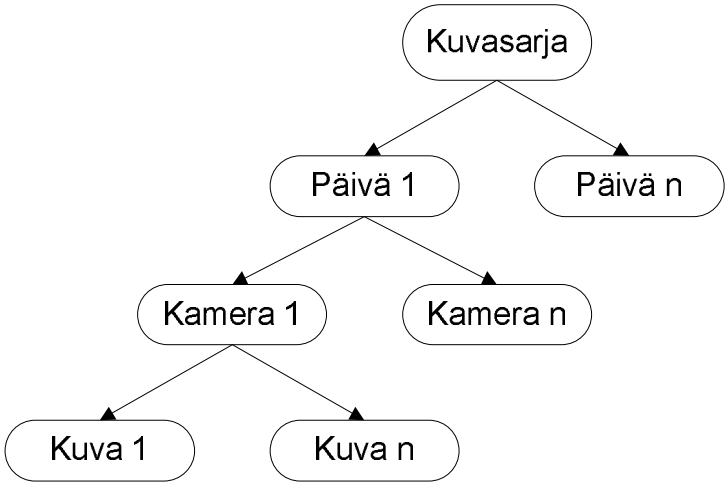
Kuvien kooksi tuli niiden sisällöstä riippuen 10 – 15 megatavua per kuva. Testeissä keskimääräinen kuvakoko oli noin 12 megatavua ja tällä tiedostokoolla arvioitiin kuvakapasiteetiksi yli 300 000 kuvaa, mikä vastaa 60 000 kuvaa per kamera. Kolmenkymmenen sekunnin kuvavälillä tämä vastaisi kolmen viikon kuvausaikaa. Mittauksen aikana päädyttiin kuitenkin vaihtoehtoon, jossa käytettiin 60 s kuvaväliä aktiivisten mittausten aikana ja tunnin kuvaväliä mittaustaukojen kohdalla. Pidempää kuvaväliä käyttämällä saavutettiin kaikki tutkimukset kattava tallennuskapasiteetti.

Varsinaisen kuvien tallennuksen suoritti tallennusta varten laadittu ohjelma, joka käytti IDS-imaging GmbH:n kameroiden hallintaan tarjoamaa ohjelmarajapintaa. Rajapinnan kautta ohjelmalla pystyy käyttämään kaikkia kameroiden ominaisuuksia ja hallitsemaan kuvien siirtoa koneelle. Ohjelma pysäyttää kaikkien kameroiden kuvat samanaikaisesti ja tallentaa ne hakemistorakenteen ja tiedostonimen määräämään muotoon, joilla kuvat erotellaan eri kameroiden kuviksi.

4.6 Kuvien tallennus

Kuvat tallennettiin mittausten aikana hakemistorakenteeseen, jossa kuvat oli jaoteltu tallennusajankohdan ja kameran nimen perusteella. Hakemistojen ja tiedostojen aikaleimat tallennettiin kansioihin ja tiedostonimiin kuvan 4.4 puurakenteen mukaisesti. Hakemistoihin lisättiin kokonainen aikaleima ja tiedostonimiin siirros hakemiston ajankohdasta. Kuvien nimiin lisättiin vielä juokseva numerointi mahdollisen sekaantumisen estämiseksi sekä indeksointia varten.

Kuvat olisi voinut tallentaa myös tietokantaan, mutta tässä yhteydessä kuviin haluttiin päästä käsiksi ilman tietokantajärjestelmää. Ilman tietokantaa kuvien haku oli hieman monimutkaista. Tietojen hakuun kehitettiin ohjelmakoodiluokka, jonka kautta kuviin pääsee käsiksi aikatietojen perusteella.



Kuva 4.4. *Kuvien tallentamiseen käytetyn hakemistopuun rakenne.*

Hakemiston nimirakenne päiväjaossa on ”Splittime_2010-07-12T000037”, jossa ”Splittime” osoittaa kansion olevan mittauksen osa ja loppu on aikaleima tiedostojärjestelmään sopivassa ”vuosi – kuukausi – päivä T tunti – minuutti - sekunti” kirjoitusmuodossa. Kuvan tiedostonimi ”37048_uEye_1_3140” muodostuu aikaerosta sekunteina hakemiston luontiaikaan, keskellä on kameran nimi, jolla kuva on otettu ja lopussa on yksilöllinen numerointi.

Ennen kuvien käsittelyä kuvatiedostoista lasketaan indeksikartat. Indeksikartassa on yhden kameran kuvat aika-tiedostonimi parien muodossa, jolloin kuvien haku käsittelyyn saadaan helpoksi, monimutkaisesta hakemistorakenteesta riippumatta. Indeksi lasketaan ennen kuvien käsittelyä kerran, jolloin tiedostoja ei tarvitse käydä läpi useampaan kertaan. Muutaman sadan tuhannen tiedoston indeksoiminen levyltä kestää lähinnä hakemistojen selauksen vuoksi kohtuullisen pitkään, joten indeksointi on varsin järkevää ja kuvien haku on helppoa pelkän ajan perusteella.

4.6 Pakkausformaattien valinta

Pakkausformaattien pääsääntöisenä valintaperusteena oli mahdollisimman hyvä kuvanlaatu. Formaatin valinta oli vapaampaa, koska kuvien avaamiseen ja käsittelyyn käytettiin omia ohjelmia, jolloin voitiin käyttää formaatteja, joita ei tarvitse pystyä avaamaan loppukäyttäjän toimesta. Videotiedoston pakkaamiseen pyrittiin valitsemaan mahdollisimman hyvälaatuinen, mutta kuitenkin hyvin tuettu formaatti.

Pakkaamaton yksittäinen kuva veisi noin 30,2 megatavua. Viiden kameran tilatarpeeksi jokaiselle kuvaushetkelle olisi näin ollen tullut noin 151 megatavua. Pakkaamalla kuvat häviöttömästi saatiin viiden kameran kuvat mahtumaan noin 60 megatavuun. Lopputulokseksi saatavan videotiedoston kokoon ei otettu juurikaan kantaa, koska kokoon vaikuttavat enemmänkin videon kuva-ala ja kestoaika.

Paras kuvaformaatti kuvanlaadun kannalta olisi ollut kameran tuottama RAW-kuva, mutta sen käyttäminen olisi vienyt huomattavasti enemmän levytilaa ja kyseiselle kamerakohtaiselle formaatille ei ole tehokasta pakkausmenetelmää valmiiksi saatavilla.

RAW-kuvassa kameran kennon tuottama kuva on interpoloimattomassa muodossa, joten sitä on käsitelty mahdollisimman vähän ja siitä olisi ollut teoriassa mahdollista saada kuvankäsittelyllä hieman terävämpiä kuvia, käyttämällä interpolointiin kehittyneempiä menetelmiä.

4.6.1 Kuvanpakkausformaatti

Kuvanpakkausformaatiksi valittiin luvussa 3.1 esitelty HD Photo formaatti. HD Photo formaatilla häviöttömästi pakattujen kuvien koko oli testien perusteella 10 – 15 megatavua. Kuvien koko vaihteli kuvan monimutkaisuuden mukaan. Pakkausformaatiksi päädyttiin valitsemaan häviötön pakkausmuoto, koska varsinkin erotuskuvien laskennan yhteydessä esimerkiksi JPEG-formaatilla lohkojen reunat olisivat voineet muodostaa näkyviä häiriöitä.

HD Photo formaatti valittiin käyttöön sen hyvien pakkausominaisuuksien johdosta varsinkin kuvien häviöttömän pakkauksen osalta. Projektin toteutushetkellä oli saatavilla teratavuluokan tallennusjärjestelmiä suhteellisen edulliseen hintaan, joka mahdollisti kameroiden kuvadatan tallentamisen häviöttömässä muodossa, jolloin kuvien häviöllisestä pakkauksesta aiheutuva kohina ei näkyisi lopputuloksessa. Formaatissa oli myös mahdollista muuttaa tarvittaessa häviötön pakkaus häviölliseksi pelkällä parametrimuunnoksella, portaattomasti kuvanlaadun suhteen.

HD Photo formaatin huonoja puolia on rajoitettu käytettävyys. Ohjelmia, jotka osaavat käsitellä kuvia tässä formaatissa on tarjolla rajallisesti. Tässä yhteydessä kuitenkin käytetään valmista kirjastoa, joka tallennetaan ohjelman yhteyteen, joten kuvien käsittelyyn tällä ei ole vaikutusta. Kuvat ovat joka tapauksessa tarvittaessa avattavissa vapaassa levityksessä olevilla ohjelmistoilla. Toinen huono puoli HD Photo formaatissa oli että kuvien lataamiseen ja tallentamiseen ei ollut saatavilla useita ohjelmakirjastoja. Kuvien lataamiseen käytetty Microsoftin valmistama ohjelmakirjasto oli suhteellisen hidas ja muodostui lopulta kuvien käsittelyn hitaimmaksi kohdaksi.

Ohjelmissa kuvien pakkaamiseen ja purkamiseen käytetty ohjelmakirjasto oli Microsoftin tekemä HD Photo Device Porting Kit –kirjasto. Kirjastosta löytyy toteutukset erilaisille HD Photo rajapinnoille, mutta koodi ei ollut täysin yhteensopivaa käytetyn MinGW-kääntäjän kanssa, joten kirjastolle tehtiin uusi dll-rajapinta, joka toimi MinGW-kääntäjässä.

4.6.2 Videoformaatti

Kappaleessa 3.2 esitelty MPEG-4 video tallennettiin AVI-tiedostoformaatissa, koska FFmpeg-ohjelmakirjastosta videota ei saanut ulos toimivana versiona muissa säiliötyypeissä tai helposti useimmilla toisto-ohjelmilla toistettavassa muodossa. AVI-tiedosto on multimedian tallentamiseen tarkoitettu säiliötyyppinen tiedosto, joka kehystää erilaiset multimediaan liittyvät tiedot toistoa varten sopiviin paketteihin.

Formaattia valittaessa ei äänen pakkausformaattiin tarvinnut ottaa kantaa, koska videotiedostoon ei tule ääniraitaa mukaan. Xvid formaatti valittiin käyttöön, koska kuvanlaatu on riittävän hyvä tarkoista kuvista muodostettuun videoon, vaikkakin käytetty YUV 4:2:0 värikoodaus huonontaa värien tarkkuutta. Tämä ei ole juurikaan ongelma, koska kuvadata on pääosin harmaata ja väri-informaation huonompi resoluutio ei näy videolla.

Kuvadatan luonteen ollessa erittäin staattista pakkaussuhteella ei juuri ole merkitystä. Käytännössä koodekki ei edes käytä kaikkia yksittäiselle kuvalle varattuja bittejä suuremmilla bittinopeuksilla. Videoformaatin etuna on myös videotiedoston pieni koko suurilla resoluutioilla. Hyvän kuvanlaadun lisäksi formaatin valintaperusteina oli muihin hyvälaatuisiin koodekkeihin verrattuna formaatin laajamittainen tuki eri toisto-ohjelmissa.

Videotiedostojen tekoon käytetty FFmpeg-ohjelmakirjasto sisältää erilaisia kaupallisesti vapaasti käytettäviä video- ja äänenpakkauskoodekkeja sekä muita videon käsittelyyn tarvittavia työkaluja. Video- ja audiokoodekit ovat FFmpeg-ohjelmakirjastoon kuuluvassa libavcodec-kirjastossa. Ohjelmakirjastoa pystyy käyttämään suoraan koodista, kunhan kirjasto on ensin käännetty MinGW-kääntäjälle sopivaan muotoon.

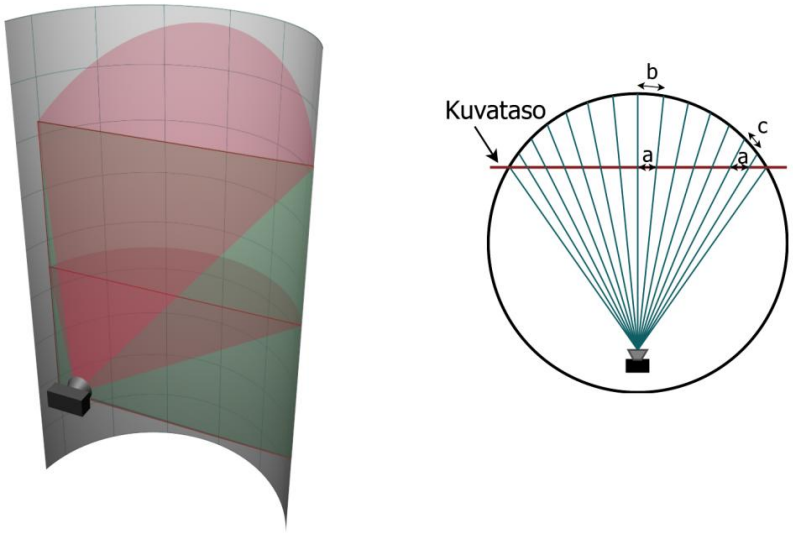
4.7 Kuvien käsittely

Kuvien käsittelyyn laadittiin erillinen ohjelmisto, jonka tarkoituksena oli muodostaa kuvasarjoista videotiedosto. Ennen kuin kuvat lisättiin videotiedostoon, täytyi niille tehdä kuvankäsittelyn avulla erilaisia muunnoksia. Kuvien oikaisut on toteutettu omilla laskenta-algoritmeilla näytönohjaimen laskemana OpenGL-rajapinnan kautta ja tavallisemmat, vakiintuneessa käytössä olevat kuvankäsittelyalgoritmit ovat ImageMagick-ohjelmakirjaston tarjoamia toteutuksia.

Kallion pintaa kuvasi yhteensä viisi eri kameraa, joiden kuvat tuli lopputuloksessa yhdistää. Kuvat otettiin niin että niissä oli päällekkäisyyttä, joten kuvia pystytään sekoittamaan päällekkäisiltä osin ja näin häivyttämään kuvien välille muutoin muodostuva raja. Kuvien oikaiseminen ja yhdistäminen helpottaa muutoksien tarkastelua, sekä parantaa kuvamateriaalin visuaalista havainnollisuutta. Kameroiden väliset kuvat pyrittiinkin yhdistämään toisiinsa mahdollisimman saumattomasti, jolloin huomio ei kiinnity kuvien kuvaamistavasta johtuviin virheisiin.

Tasaista pintaa kuvattaessa kameroiden kuvat olisi voinut yhdistää suoraan toisiinsa, mutta kuvattaessa sylinterinmuotoisen kappaleen pintaa sisäpuolelta kuvat ovat venyneet laidoilla eri tavoilla ja vaakasuuntainen pikselitiheys per millimetri ei ole vakio. Koska kameran kuvakenno on taso ja kuvattava kohde ei ole tasopinta, osassa kuvaa kuvattava sylinterin sisäpinta on lähempänä kameraa, kuin muut kohdat. Tämä aiheuttaa kahdenlaista muotovirhettä kuvan 4.5 esimerkkitapauksessa. Kuva on suurentunut keskikohdalta ja pystysuunnassa kuva-alue ei pysy neliömäisenä. Käytännössä ongelma on monimutkaisempi jos kamera ei ole täsmälleen suorassa.

Muotovirheen korjaus toteutettiin ohjelmassa kappaleessa 3.10 esiteltyjen tekstuuriprojektioiden avulla tapahtuvien muunnosten kautta.



Kuva 4.5. *Kuvan vääristymä tasokuvaan nähden, sylinterin sisäpintaa kuvattaessa.*

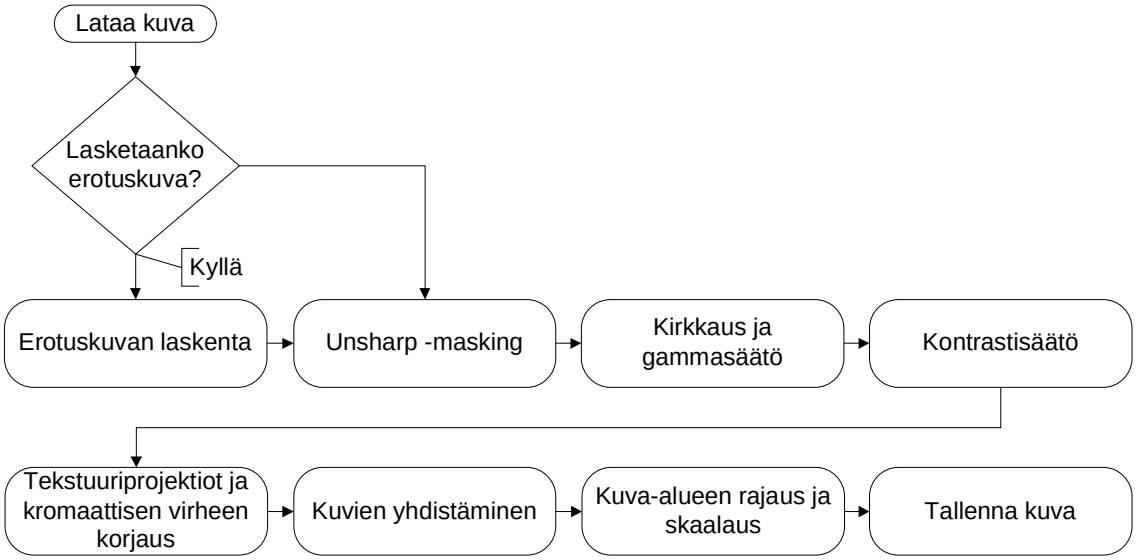
Kuvassa 4.5 on näkyvillä, kuinka sylinterin sisäpinnalta otettu kuva poikkeaa tasopinnasta otetusta kuvasta. Kuvan vasemmalla puolella olevasta läpinäkyvästä vihreästä tasosta voidaan nähdä, kuinka paljon sylinterin pinnalta otettu kuva poikkeaa sen kattamasta kuva-alasta eri horisontaali-tasoissa. Oikealla puolella olevasta kuvasta näkyy, kuinka vaakasuunnassa pikselitiheys tulee erilaiseksi verrattuna tasokuvaan. Kuvassa välit a ovat tasolla samanmittaiset, mutta sylinterin pinnalla välit ovat erilaiset ja vielä niin että väli $b > c$, jolloin pikselitiheys ei ole sylinterin pinnalla vakio.

Käytännössä korjaus ei ole edes niin yksinkertainen kuin kuvassa 4.5, vaan kamerat osoittavat hieman eri suuntiin ja ovat esimerkiksi hieman kiertyneet akselinsa ympäri. Myöhemmässä vaiheessa kamerat käännettiin osoittamaan vielä sivuun, jossa on mahdollisesti analyysin kannalta mielenkiintoinen halkeama. Kameroita ei yritettykään suunnata tarkasti, vaan kameroiden kuvat kohdistetaan kuvankäsittelyn yhteydessä uudelleen, jolloin ne saadaan korjattua päällekkäisiksi. Kameroiden kohdistus tarkasti olisi ollut turhan haastavaa kuvauskohteessa, jossa ollaan turvaköysien ja tikapuiden varassa.

Geometrian korjauksen lisäksi kuvien kirkkaus, kontrasti ja terävyys säädettiin mahdollisimman hyväksi ja tasaiseksi eri kuvien välillä. Tavallisen videokuvan lisäksi ohjelmassa on vielä erikseen toiminto erotuskuvien laskemiseen, jolla nähdään helpommin aikavälillä tapahtuneet muutokset kallion rakenteessa.

4.6.1 Kuvankäsittelyketju

Parhaan kuvanlaadun saavuttamiseksi kokeiltiin erilaisia kuvankäsittelyalgoritmien laskentajärjestyksiä, jonka perusteella löydettiin lopputuloksen kannalta paras kuvanlaatu. Testien perusteella oli järkevintä aloittaa kuvankäsittely terävöittämällä kuva kappaleessa 3.3 esitellyllä Unsharp masking -tekniikalla. Tähän tulokseen olisi voinut yhtä hyvin päätyä myös sitä kautta, että alkuperäisellä muokkaamattomalla kuvalla on paras mahdollinen tarkkuus, koska jotkin kuvaan kohdistuvat kuvankäsittelyoperaatiot tekevät vierekkäisistä pisteistä toisistaan riippuvaisia. Lisäksi osa algoritmeista aiheuttaa myös eri väritasojen välisiä riippuvuussuhteita kuvapisteiden väriarvoihin. Näistä esimerkkeinä ovat kuvan sumentaminen, joka tekee vierekkäisistä pisteistä toisistaan riippuvia tai väriavaruuden muunnos RGB:stä YUV- muotoon luo riippuvuuden väritasojen välille.



Kuva 4.6. *Kuvien käsittelyalgoritmien käyttöjärjestys.*

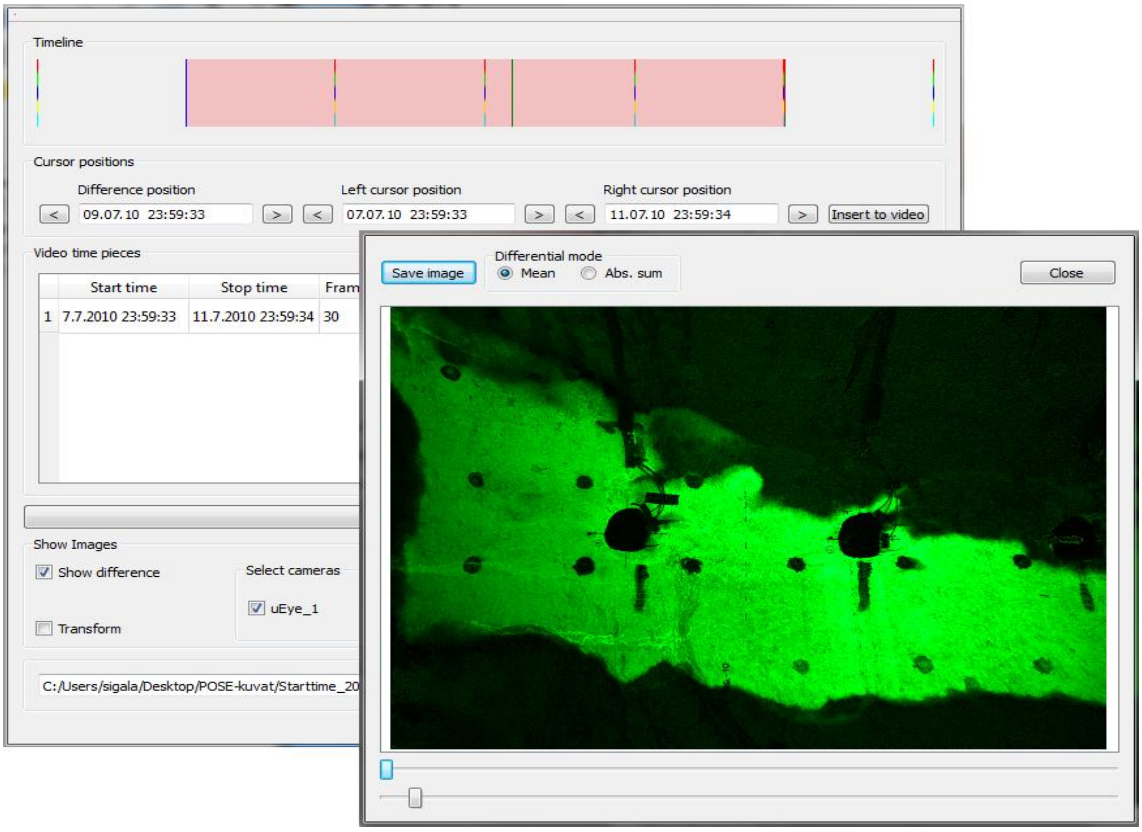
Kaikki alkuperäiseen kuvaan kohdistuvat operaatiot tähtäävät ainoastaan visuaalisen havainnollisuuden parantamiseen. Koska alkuperäisellä kuvalla on paras muokkaamaton resoluutio, kuvan terävyyttä parantavat suodattimet voivat tässä vaiheessa toimia parhaalla mahdollisella tavalla. Kuvien kontrasti ja kirkkaus säädettiin ennen kuvaan kohdistuvia oikaisulaskentoja, jolloin kuvapisteet ovat mahdollisimman vähän käsiteltyjä.

Osa ketjussa olevista algoritmeista voidaan myös jättää käyttämättä asettamalla sellaiset parametrit, joilla kuviin ei tapahdu muutoksia. Parametrien arvot, joilla muutosta ei tapahdu on tarkastettu koodissa, jolloin turhaa laskentaa ei tehdä, mutta vuokaaviosta ne on jätetty pois selkeyden vuoksi. Kuvassa 4.6 näkyy kuvaan kohdistuvien algoritmien käyttöjärjestys.

4.6.2 Muutosten havainnollistaminen

Kalliossa tapahtuvien muutosten havainnollistamiseksi kuvien analyysivaiheessa eri ajankohtina otettujen kuvien välillä laskettiin erotus. Erotuskuvassa näkyvät melko pienetkin muutokset. Erotuskuvat laskettiin kappaleessa 3.5 esitellyllä menetelmällä.

Erotuskuvan käyttöä rajoittavat lämpölaajenemisen ja muiden tekijöiden aiheuttamat muutokset kameran kiinnityspisteiden kohdistuksissa pitkillä aikaväleillä. Lisäksi pienimpien muutosten osalta kameran kuvassa oleva kohina rajoittaa niiden vahvistamista. Käytännössä kuvassa tapahtuvat muutokset ovat kuitenkin varsin suuria esimerkiksi kun valaistusolosuhteet ja valojen aiheuttamat varjot muuttuvat. Valaistusolosuhteet voivat muuttua esimerkiksi kosteuden tummentaessa kiviainesta.

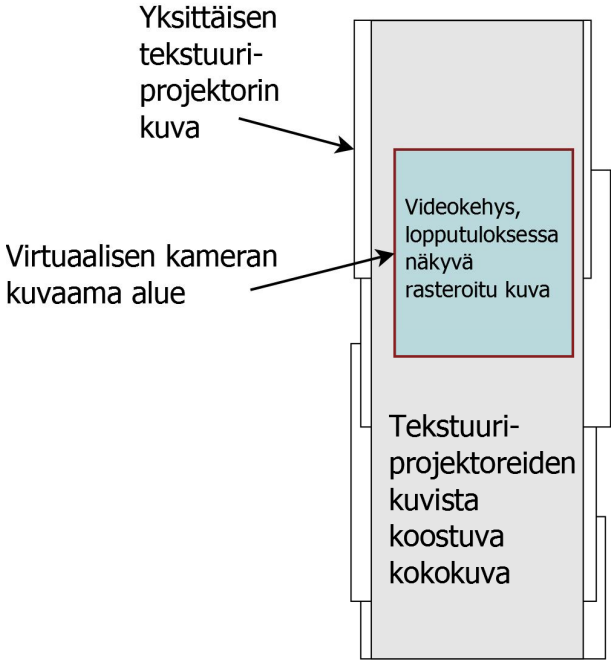


Kuva 4.7. *Erotuskuvan laskentakohdan valinta toteutetussa ohjelmassa. Kuvan alemmassa osassa näkyy kosteuden aiheuttama muutos kahden kuvan välillä, joka on laskettu käyttäjän valitsemilla ajan kohdilla.*

Ohjelmassa erotuskuvien laskenta toteutetaan kiinnittämällä vertailukuva vihreällä kursorilla, joka näkyy kuvassa 4.7 ylälaidassa punaisen alueen keskikohdassa. Käyttäjä valitsee kuvasarjan aikajanalta kohdan, johon kaikki muut kuvat vertaillaan ja videotiedostoon tallennetaan kuvista lasketut erotukset. Kuvassa näkyvä punainen väli aikajanalla on käyttäjän valitsema alue, josta video muodostetaan.

4.6.3 Tutkittavan alueen rajaus ja skaalaus

Lopputuloksen kannalta on oleellista määritellä tutkittava kohta tarkemmin. Jos koko kuva-ala tallennettaisiin täydellä resoluutiolla videotiedostoon sen resoluutioksi tulisi 2748 x 12000 pikseliä, jolloin videon tallentaminen ja toisto ei olisi kovinkaan käytännöllistä. Kuvassa 4.8 on esimerkkinä valittu tutkittavaksi alueeksi osa usean kameran muodostamalta kuva-alalta. Virtuaalisen kameran avulla tutkittava alue voidaan siirtää, skaalata ja kiertää haluttuun kohtaan kolmiulotteisen mallin ympäristössä. Lisäksi näkymä voidaan muuttaa perspektiivinäkymäksi, jolloin voidaan tutkia kuvaa alkuperäisen muodon mukaisessa näkymässä.



Kuva 4.8. *Tutkittavan alueen rajaus kameroiden kuvien yhdistelmästä.*

Videotiedostoon tallennettava kohta määritellään ohjelmassa parametreilla. Videoon tallennettava alue annetaan ohjelmalle kehyksenä, jossa on kehyksen kulmapiste alkuperäisessä, koko alueen kattavassa kuvassa. Toisena parametrina ohjelmaan annetaan videotiedoston kuva-alueen koko, jolloin ohjelma laskee tarvittavan skaalauksen. Kuvattava kohta määritellään virtuaalisen kameran paikkakoordinaateilla ja suuntakulmilla.

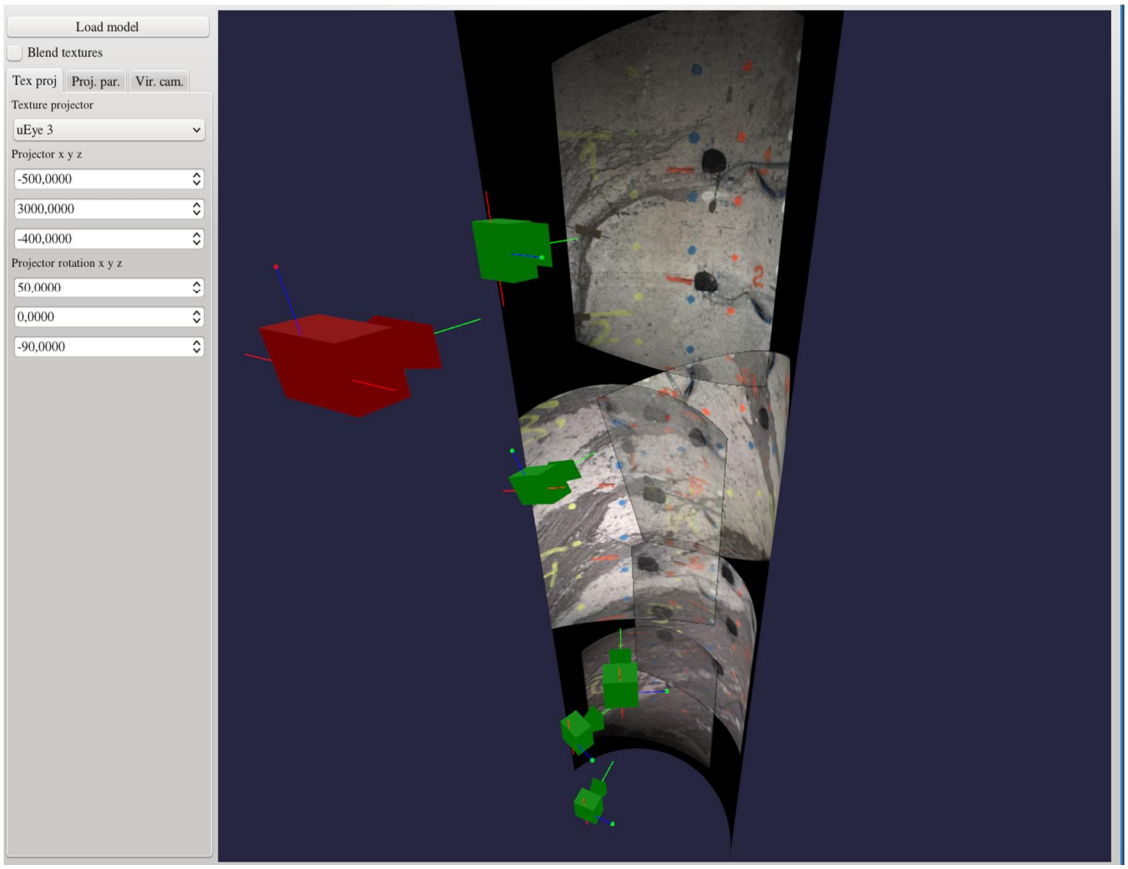
4.6.4 Virtuaalikameran asettelu ja videokehys

Lopputuloksessa näytetään eri kameroiden kuvista koostetusta kuvasta jokin käyttäjän valitsema alue. Videotiedostoon tai ruudulle päätyvä kuva muodostetaan OpenGL:n avulla. Tulos lasketaan virtuaalisen kameran avulla, joka sijoitetaan mallinnetun kohteen ympäristöön haluttuun kohtaan. Kameran projektio voidaan valita joko ortogonaaliseksi tai perspektiiviseksi.

Kameran kuvaama alue määritellään ortogonaaliprojektiossa kuvattavan mallin millimetriskaalassa olevalla aluerajauksella ja perspektiiviprojektiossa kameran FOV:n mukaan. Virtuaalikameran kuvaama alue määritetään 3D avaruudessa asettamalla kameran paikka ja suuntakulma. Jotta saataisiin halutun kokoinen rasterikuva, täytyy 3D avaruudessa määritelty kuva-ala pienentää tai suurentaa halutun kokoiseksi. Ortogonaali- ja perspektiiviprojektion tapauksessa lopputuloksen koko, johon kuvattava alue tallennetaan, määritetään m, n kokoisena kuva-alueena. Skaalaus tapahtuu OpenGL:n toimesta laskennan yhteydessä.

Ortogonaaliprojektio tuottaa kuvaksi levymäisen pinnan ilman perspektiiviä, jolloin kuvatun kohteen muodot litistyvät kuvatasolle ja muodostavat samankaltaisen skaalan kaikille kuvassa näkyville etäisyyksille. Perspektiiviprojektio tuottaa perspektiivinäkymän, jossa näkyy kuvattavan kohteen kolmiulotteiset muodot, mutta etäisyydet eivät ole enää samassa skaalassa.

Kameran kohdistaminen haluttuun pisteeseen ja lopputuloksena olevan kuvan kierto onnistuu muuttamalla virtuaalisen kameran sijaintia ja suuntakulmia. Kuvassa 4.9 näkyvät tekstuuriprojektorit, vihreät laatikot, jotka ovat alkuperäisten kameroiden paikoissa ja virtuaalinen kamera, punainen laatikko, jonka kuvaamaa aluetta tarkastellaan toisessa näkymässä ja joka tulee näkymään lopputuloksessa.



Kuva 4.9. *Tekstuuriprojektoreiden sijoittelunäkymä toteutetussa ohjelmassa.*

Ennen kuin videokehys tallennetaan videotiedostoon, siihen lisätään haluttaessa kuvan tallennusajankohtaan liittyvä aikaleima, jolloin kyseistä ajanhetkeä voidaan tarvittaessa tarkastella tarkemmin myöhemmän analyysin yhteydessä. Aikaleimat on määritetty kuvien tallennusajankohtien mukaisesti.

4.7 Videotiedoston muodostaminen

Ohjelmassa lasketut kuvat muutetaan videotiedostoksi FFmpeg-ohjelmakirjastolla. Videotiedoston muodostamista varten täytyy myös hakea kuvien tallennusajat, jotta osataan hakea oikeat kuvat oikeasta kohdasta videolle. Luvussa 4.6 on esitelty kuvien tallennusmenetelmä, josta tiedostot ja ajat indeksoidaan, kuvien hakua varten.

Aikojen hakua varten kaikki kuvat on aluksi indeksoitu karttatyyppiseen tietorakenteeseen T, jossa on $< tiedostonimi, aika >$ tietopari K. Tietoparin K sisältävät kartat T on vielä tallennettu erilliseen $< kameranimi, T >$-karttaan, jolloin voidaan muodostaa useamman kameran kuvien indeksit, joiden tietojen ei tarvitse olla täysin yhtenevät. Indeksi lasketaan vain tarvittaessa uudelleen, koska sen muodostaminen kestää melko pitkään satojen tuhansien tiedostonimien käsittelyn takia.

Videoon tallennettavat kuvat saadaan haettua kartasta nopeasti lähimmän osuvan ajankohdan perusteella. Jos täydellistä osumaa ajankohdalle ei ole, palautetaan lähin vastaava kuva. Videon toistoväli valitaan antamalla alku ja loppuajat videon kuvasarjan aikajanalta.

Videossa lopputuloksen kannalta on oleellista, kuinka nopeasti kuvasarja toistetaan. Toistonopeutta kontrolloidaan kahdella parametrilla, kuvien väli sekunteina ja videon kuvataajuus. Näillä kahdella parametrilla saadaan videon toistoa nopeutettua tai hidastettua. Hidastuksessa toistetaan samaa kehystä n kertaa ja nopeutuksessa ohitetaan n kehystä kerralla. Kuvien väli sekunteina -parametri kertoo, kuinka monta sekuntia per kuva edetään reaalimaailman ajassa, jonka mukaisesti kuvasarja on tallennettu. Kuvataajuus taas kertoo, montako kuvaa videoon tallennetaan sekunnissa.

Varsinaisen videotiedoston muodostaminen ffmpeg-ohjelmakirjastolla on yksinkertaista. Alustetaan koodekki tarvittavilla parametreilla, syötetään koodekille n kehystä koodattavaksi ja lopuksi suljetaan videotiedosto. Videokoodekki alustettiin syöttämällä videotiedoston formaatin parametrit. Formaatteina käytettiin kappaleessa 3.2 esitettyä Xvid MPEG-4 koodekkia ja ääniformaatiksi ei asetettu mitään.

4.8 Laskentatehokkuus

Kuvien käsittely on laskentaintensiivinen operaatio, joten olisi hyvä löytää mahdollisimman nopea käsittelytapa kuville, jotka tallennetaan videotiedostoksi. Jos esimerkiksi koko tallennuskapasiteetin 60000 kuvaa muutettaisiin videoksi ja jokaisen kuvaushetken kuvien käsittely kestäisi 5 sekuntia, menisi videotiedoston tekoon noin kolme ja puoli päivää. Hyvän kuvankäsittelynopeuden saavuttamiseksi on käytetty erilaisia keinoja.

Tehokkain keino laskentatarpeen vähentämiseksi on ollut rajata turhat laskutoimitukset pois laskennasta. Esimerkiksi jos kameran kuva ei näy lopputuloksessa, ei kyseisen kameran kuvaa tarvitse laskea ollenkaan. Seuraavaksi laskenta-aikaa lyhennettiin etsimällä kaikkein aika-intensiivisimmät kohdat ohjelmasta ja optimoimalla ne. Hitaimmaksi yksittäiseksi kohdaksi kuvienkäsittelyssä jäi kuvien lataaminen levyltä.

Ohjelmassa on käytetty rinnakkaislaskentaa mahdollisuuksien mukaan. ImageMagick-ohjelmakirjaston tarjoamat algoritmit tukivat jo valmiiksi rinnakkaislaskentaa. Omat algoritmit muunnettiin rinnakkaislaskentaa käyttäväksi joko pilkkomalla kuvankäsittelyoperaatio rinnakkaislaskettavaksi tai jos tämä ei ollut mahdollista, eri kuvia käsiteltiin samanaikaisesti eri säikeissä.

Käytännössä rinnakkaislaskennan toteuttaminen oli näissä tapauksissa yksinkertaista. Käytetyssä C++ -kääntäjässä oli tuki OpenMP API:lle, jonka avulla rinnakkaislaskennan käyttöönottaminen oli käytännössä vain muutaman rivin lisääminen ohjelmasilmukoiden ympärille. Rinnakkaislaskennan käytössä oli otettava huomioon, että laskentaongelma oli jaettavissa säikeiden kesken. OpenMP määritelmän [9] mukaan OpenMP ei ota kantaa laskentatulosten oikeellisuuteen, vaan jättää vastuun ohjelmakoodin kirjoittajalle.

Kuvankäsittelyssä melkein kaikki algoritmit ovat sellaisia, joissa vierekkäiset pisteet voidaan käsitellä erikseen. Yleensä monissa laskentaongelmissa tarvitaan edellisen laskentapisteen tulos, jota muokataan iteratiivisesti. Tällöin ongelma ei yleensä ole rinnakkain laskettavissa. Kuvankäsittelyssä käytettävissä laskenta-algoritmeissa on yleensä jokin lähtödata, mikä muunnetaan suoraan lopputulokseksi, muista tuloksista riippumatta. Tällaiset ongelmat ovat hyvin helposti muutettavissa rinnakkaislaskentaa käyttäviksi.

Rinnakkaislaskennan lisäksi hyödynnettiin kuvien oikaisussa ja yhdistämisessä OpenGL-rajapintaa, jonka avulla voidaan käyttää näytönohjaimen kuvankäsittelyyn suunnattua rautapohjaista prosessointia. Kuvien oikaisujen laskennassa tarvittavat projektiot ja muunnokset ovat nopeita laskea tarkoituksenmukaisella laitteistolla, koska laskentaongelma on jaettavissa eri laskentayksikköjen kesken. Tämän dokumentin kirjoitushetkellä halvoissa näytönohjaimissa on kymmeniä laskentayksikköjä ja kalliimmissa satoja, kun PC prosessorissa yleisimmin on 1-4 tehokasta yleiskäyttöistä prosessoria.

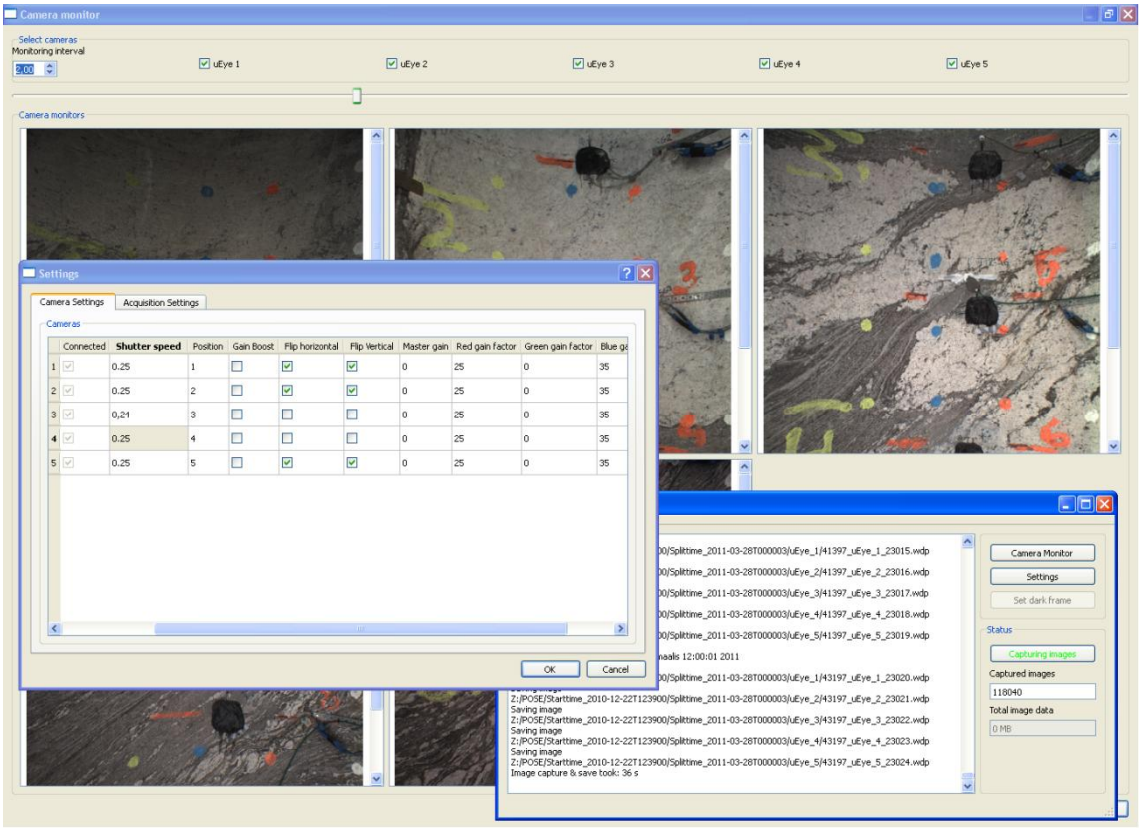
4.9 Ohjelmien toteutus

Ohjelmien suunnittelu aloitettiin keräämällä tarvittavat tiedot ohjelmien toiminnallisuuksista. Kerättyjen tietojen perusteella kuvien tallennusta ja käsittelyä varten päädyttiin laatimaan erilliset ohjelmat, koska oli nähtävissä että projektissa tarvittavien ohjelmien luonne ja käyttöajankohta poikkesivat toisistaan. Projektin edistymisen kannalta kuvien tallennusohjelmiston tuli olla valmiina aiemmin.

Varsinaiset toteutukset tehtiin C++ ohjelmointikieltä ja Qt SDK-työkalua käyttäen. Käyttöliittymien ja ohjelmakoodin toteutukseen sekä ohjelmakoodin kirjoittamiseen käytettiin Qt Creator ohjelmointiympäristöä. Ohjelmien kuvankäsittelyalgoritmien testaukseen ja laatimiseen käytettiin Matlab-ohjelmistoa. Matlabilla laaditut algoritmit muutettiin myöhemmin C++ koodiksi.

4.9.1 Tallennusohjelma

Tallennusohjelman käyttötarkoituksena on ajoittaa kuvien tallennushetket, tallentaa kuvat kameroilta ja nimetä kuvat, niin että ne voidaan jälkeenpäin yhdistää halutunlaiseksi lopputulokseksi. Lisäksi ohjelmalla voidaan monitoroida kuvattavan kohteen tilaa tutkimuksen aikana. Ennen kuvien tallentamisen aloittamista kameroiden valotukset ja analogiset vahvistimet yms. asetetaan kohdilleen ohjelman kautta.

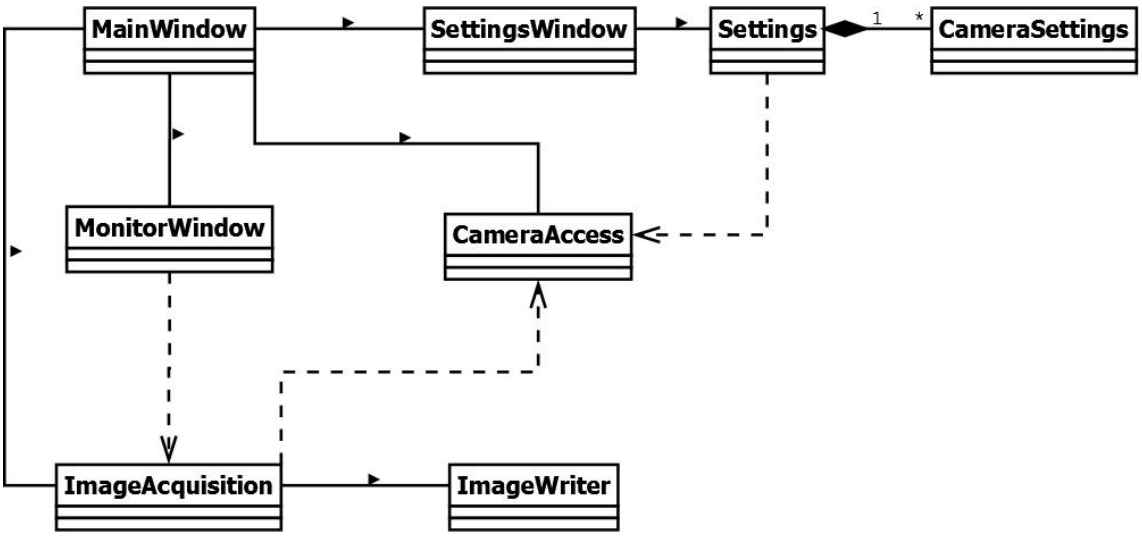


Kuva 4.10. *Kuvankaappaus kuvien tallentamiseen tehdystä ohjelmasta.*

Kuvassa 4.10 näkyvässä Settings-ikkunassa voidaan asettaa kameroille tarvittavat parametrit. Jokaiselle kameralle voidaan asettaa erikseen analogisten vahvistimien vahvistuskertoimet, sulkimen nopeus ja kameran kuvan käännöt, jos kamera on sijoitettu väärin päin suhteessa kohteeseen. Yleisinä parametreina kaikille kameroille voidaan asettaa kuvien kuvausväli ja tallennuskansiot. Camera monitor -ikkunasta voidaan valvoa kameroiden ottamia kuvia mittauksen aikana. Kuvat päivitetään aina kun kamerat eivät tallenna täyden resoluution kuvia, jottei kuvien tallennus häiriintyisi.

Kuvien tallennusohjelma koostuu kahdessa eri säikeessä ajettavasta prosessista. Ohjelman pääsäie suorittaa käyttöliittymään ja ohjelman yleiseen toimintaan liittyvät toiminnot. Pääsäikeessä suoritettavia toimintoja ovat asetusikkunat, tallennusloki ja kameroiden valvontaikkuna. Kuvan 4.11 luokkakaaviosta selviää ohjelman sisäinen rakenne.

MainWindow-luokka on ohjelman käyttöliittymän pääikkuna ja SettingsWindow- sekä MonitorWindow-luokat ovat ohjelman asetus- ja monitorointinäkymät. Settings-luokka tallentaa yleiset asetukset asetustiedostoon ja sisältää CameraSettings-luokan tiedot, joissa on kamerakohtaiset asetukset. Lisäksi asetuksien hallinta ja muutosten välitys tapahtuu k.o. luokan kautta. CameraAccess-luokka huolehtii kaikesta kommunikaatiosta kameroiden ajurin ja ohjelman välillä. ImageAcquisition-luokka on taustasäie, joka toteuttaa kameroiden kuvien hakemiseen kohdistuvat operaatiot. ImageWriter-luokka toteuttaa kappaleessa 4.6 olevan kuvien tallentamisen hakemistorakenteeseen ja tallentaa kuvat sen mukaisesti.



Kuva 4.11. *Tallennusohjelman yksinkertaistettu luokkakaavio.*

Varsinainen kuvien tallennus suoritetaan taustasäikeessä, jonka tarkoituksena on huolehtia kaikesta kommunikaatiosta ohjelman ja kameran välillä, lukuun ottamatta kameroiden asetusten muutosta, joka voidaan suorittaa lukituksien kautta mistä tahansa säikeestä. Ohjelman käyttöliittymän ja varsinaisen työprosessin erottamisella saadaan ohjelman käytettävyys huomattavasti paremmaksi. Ohjelman käyttöliittymä on käytettävissä ohjelman hakiessa kuvia ja tallentaessa niitä. Taustasäikeen prosessi

kuuntelee viestejä pääohjelmalta ja suorittaa niitä normaalin kuvankaappausprosessin lisäksi. Näitä toimintoja ovat mm. valvontaikkunaan kaapattavat kuvat, joita otetaan, aina mahdollisuuksien mukaan, kun säie ei tallenna kuvia.

Kuva 4.12. *Kameroiden kuvien tallennusketju.*

Taustasäie käsittelee saadut viestit ja tarkkailee kelloa seuraavaan kuvanottohetkeen. Kun seuraavan kuvan tallennushetki on saavutettu, käynnistyy kuvan 4.12 mukainen kuvien tallennusoperaatio. Ennen kuvien tallentamista lasketaan seuraavan kuvan ottamisajankohta, jotta ajastus ei häiriintyisi kuvien tallentamisen ollessa vaihtelevan mittainen operaatio.

Kuvien tallennusohjelman toimintaa testattiin ennen käyttöä noin kahden kuukauden ajan mahdollisten ongelmien löytämiseksi. Muutamia vikoja löydettiinkin ja ne korjattiin. Varsinaisena ohjelman käyttöaikana ei ole ilmennyt ongelmia, jotka olisivat johtuneet itse ohjelmasta.

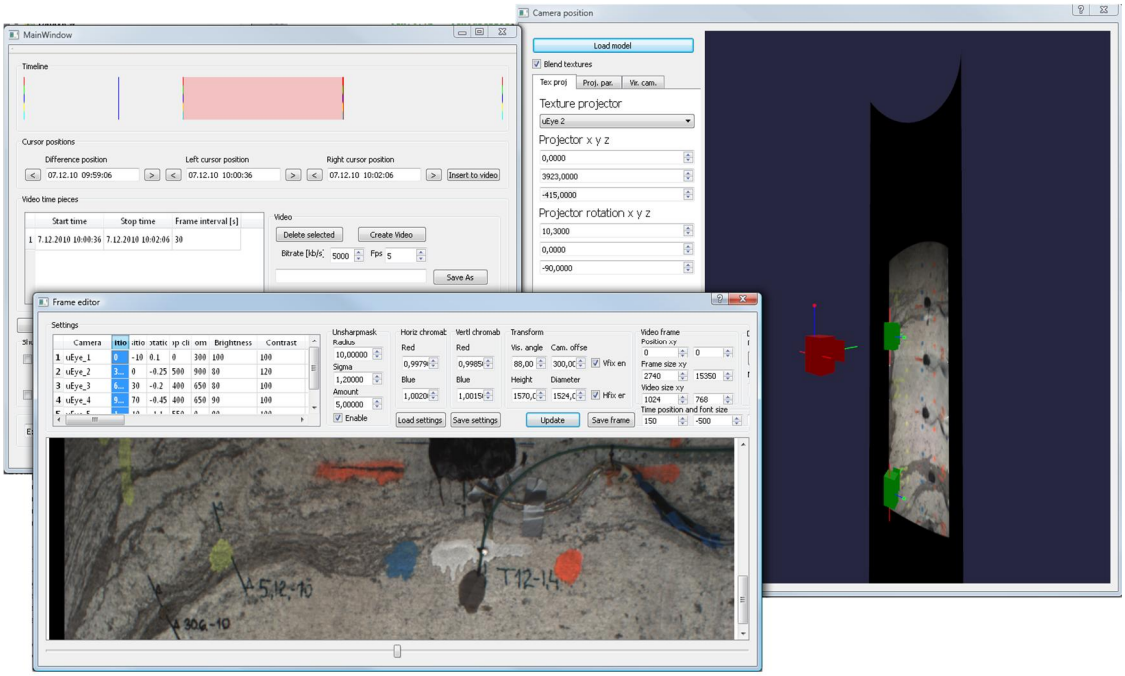
Ohjelman suunnittelussa käytettiin mahdollisimman paljon koodia, jossa muistin varaukset ja toimintalogiikka ovat suhteellisen yksinkertaisia. Muisti varattiin ohjelman käynnistyksen ja kameroiden kytkemisen yhteydessä staattisesti tai dynaamisesti ohjelman koko päälläoloajaksi, jotta ohjelman pitkän toiminta-ajan yhteydessä ei tulisi ongelmia dynaamisen muistinvarauksena kanssa. Ohjelman ollessa päällä muistia käytettiin noin 500 megatavua, eli melko runsaasti. Tässä yhteydessä haluttiin panostaa toimintavarmuuteen, joten tuhlaileva muistivarausten käyttö oli perusteltua.

4.9.2 Kuvankäsittelyohjelma

Ohjelman käyttötarkoituksena on kuvasarjojen kuvien käsittely yksittäisiksi kuviksi tai videotiedostoiksi, joita voidaan analysoida helpossa muodossa. Ohjelma indeksoi tallennusohjelmiston tallentamat kuvat aikasarjaksi, jonka jälkeen yksittäisiä kuvia tai kuvasarja voidaan valita käsiteltäväksi. Ohjelma mahdollistaa alkuperäisten ja käsiteltyjen kuvien tarkkailun, kuvien yhdistämisen, sekä videotiedoston muodostamisen halutulta alueelta ja aikajaksolta.

Kuvankäsittelyohjelmaa lähdettiin kehittämään kuvien indeksoinnista ja kuvien käsittelystä videon yksittäisiksi kehyksiksi. Aluksi kuvadatan hallintaan kehitettiin menetelmät, joilla saadaan raakadata haettua käsiteltäväksi. Sen jälkeen toteutettiin luokat, joiden avulla nähdään kuvankäsittelyalgoritmeilla saadut tulokset, jotta algoritmien kehittely olisi testattavissa. Lopuksi lisättiin varsinainen videotiedostojen tekomahdollisuus kuvasarjoista.

Ohjelmaa hallitaan kuvassa 4.13 olevien kolmen päänäkymän kautta, joiden ruuduissa voidaan muuttaa ohjelman parametreja. MainWindow-näkymässä muokataan videotiedoston aikaan liittyviä parametrejä ja Frame Editor -näkymässä puolestaan muutetaan kuvankäsittelyalgoritmien parametreja kuten kirkkaus, kontrasti ja terävöinti. Lisäksi näkymässä voi muuttaa videon kehyksen kokoa ja ominaisuuksia sekä asettaa videossa näkyvän aikaleiman sijainti. Frame editor -näkymässä on lopputulokseksi lasketun kuvan tarkasteluruutu. Camera position -näkymässä muokataan tekstuuriprojektoreiden sekä virtuaalisen kameran paikkoja ja suuntakulmia. Näkymässä olevaa 3D maailmaa voi siirrellä näppäimien ja hiiren avulla, jolloin kameroiden sijoittelua voi katsoa eri suunnilta.

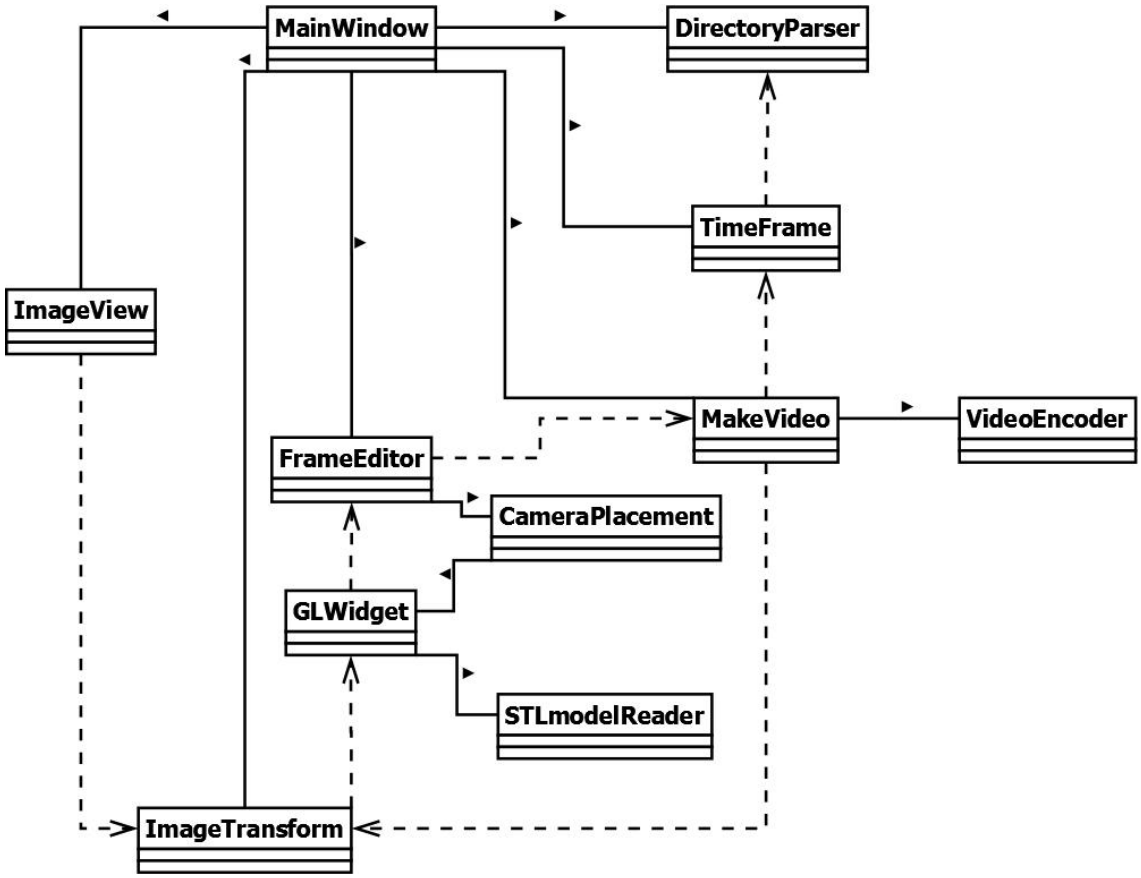


Kuva 4.13. *Kuvien käsittelyyn kehitetyn ohjelman näkymät.*

Kuvankäsittelyohjelma koostuu pääosiltaan kuvassa 4.14 olevista luokista. DirectoryParser-luokka hoitaa kuvien indeksoinnin ja TimeFrame-luokka mahdollistaa kuvien hakemisen indeksistä annetun hakumäärittelyn perusteella. MainWindow-luokka on ohjelman päänäkymä, jossa määritellään käsiteltävä ajankohta, videotiedon luontiparametrit yms. ImageView-luokka on näkymä, josta voidaan tutkia jollakin ajankohdalla olevan kuvan raakadataa yksittäisen kameran kohdalta tai kahden kuvan erotusta eri vahvistuskertoimilla.

FrameEditor-luokka on videon kehyksen määrittelyyn ja kuvan oikaisuun liittyvien parametrien hallintaan liittyvä ikkuna. MakeVideo-luokan avulla annetuista parametreista ja kuvadatasta luodaan videotiedosto kuva kerrallaan. VideoEncoder-luokka alustaa videokoodekin, luo videotiedoston, lisää yksittäiset kuvat videoon ja sulkee videotiedoston. STLmodelReader-luokka lataa STL-tiedostoformaatissa olevan kolmiulotteisen mallin. GLWidget-luokka laskee OpenGL:n avulla annettujen tietojen

perusteella kuvien oikaisun ja yhdistelmän. CameraPlacement-luokan näkymässä asetetaan virtuaalinen kamera ja tekstuuriprojektorit.



Kuva 4.14. *Kuvankäsittelyohjelman yksinkertaistettu luokkakaavio.*

Varsinainen kuvankäsittely suoritetaan pääosin ohjelmakirjastojen kautta. Kuvien käsittely ennen korjauksia suoritetaan ImageMagick-ohjelmakirjaston algoritmeilla ja muilla tarvittavilla laskenta-algoritmeilla PC:n omalla prosessorilla. Kuvien oikaisu ja yhdistäminen lasketaan OpenGL-rajapinnan kautta näytönohjaimella, josta tulokseksi saatu kuva näytetään käyttäjälle tai tallennetaan videotiedostoon.

5 JOHTOPÄÄTÖKSET

5.1 Kamerajärjestelmän toiminta

Projektia varten koottu kamerajärjestelmä toimi suunnitellulla tavalla, ainakin tämän dokumentin kirjoitushetkeen asti. Järjestelmällä saavutettiin annetun budjetin rajoissa riittävän hyvä kuvanlaatu ja toimintavarmuus.

Projektin aikana järjestelmä pysyi päällä keskeytymättä, yhtä useamman päivän kestänyttä sähkökatkoa lukuun ottamatta. Tallennusohjelmisto on toiminut moitteettomasti tallentaen kuva-aineiston varmuuden vuoksi kahdelle eri levylle. Varmuuskopiona toiminut levy tulikin lopulta tarpeeseen, koska toisesta levyjärjestelmästä hajosi yksi levy, joka ei ole mitenkään yllättävää ottaen huomioon käyttöolosuhteet. Tallennusjärjestelmän todennäköinen hajoaminen osattiin ottaa huomioon, koska olosuhteet eivät olleet kovin suotuisat elektroniikalle.

5.2 Kamerajärjestelmän spesifikaatio

Spesifikaation osalta joistakin alkuperäisistä tavoitteista jouduttiin karsimaan, koska hinta olisi noussut moninkertaiseksi, hyödyn jäädessä vähäiseksi. Esimerkiksi kameroiden kuva-alaa jouduttiin pienentämään vaakasuunnassa, jottei jouduttaisi käyttämään turhan montaa kameraa.

Kameroiden osalta käytettiin teknisistä syistä alkuperäisestä suunnitelmasta poiketen elektronisella sulkimella varustettuja kameroita, koska tavalliset kamerat olisivat mekaanisen sulkimen osalta olleet teknisesti epävarmoja. Muilta osin spesifikaation määrittelemät tavoitteet pystyttiin saavuttamaan kohtuullisen hyvin.

5.3 Kuvankäsittelyn tulokset

Projektia varten laadittu kuvankäsittelyohjelma toteutti tehtävänsä. Kuvankäsittelyn avulla saatiin kameran kuvia muokattua silmälle paremmin havaittavaksi ja kuvien yhdistäminen toisiinsa onnistui melko hyvin.

Lopputulokseksi saatuja videotiedostoja ei päästy analysoimaan, koska kuvadataa, jossa muutoksia olisi ollut, ei diplomityön valmistuessa vielä ollut saatavilla. Projekti jatkuu vielä ainakin sen aikaa, kunnes jatkotutkimuksesta saatava kuvamateriaali saadaan käsiteltäväksi ja videomuotoinen aineisto saadaan muodostettua halutuista kohdista.

5.4 Työn aihe

Diplomityön aihe oli mielenkiintoinen monipuolisuutensa vuoksi ja työhön sisältyikin lopulta kaikenlaista kuvankäsittelyyn liittyvää. Projektin alussa kameroiden ominaisuuksien ja linssien toiminnan selvittelyn yhteydessä tuli tutustuttua melko tarkasti kameroiden toimintaan. Kameroiden toimintaperiaatteiden selvittelyn lisäksi kamerajärjestelmän saamisesta toimivaksi kokonaisuudeksi oli monenlaista puuhastelua. Kameroiden kiinnitysten ja johdotusten parissa tuli vastaan kaikenlaisia pieniä teknisiä ratkaisuja vaativia ongelmia.

Projektin ohjelmointivaiheessa tuli tutustuttua hyvinkin monenlaisten ohjelmakirjastojen käyttöön ja erilaisia algoritmeja tuli kokeiltua useita ennen sopivien löytymistä. Alkuperäisen suunnitelman mukaan oli tarkoitus analysoida kerätty kuvamateriaali ja kehittää kuvankäsittelymenetelmiä kallioseinämässä tapahtuvien muutosten havaitsemiseksi. Koska POSE tutkimuksen aikataulu venyi kameravalvonta-projektista riippumattomista syistä, tämäntyyppistä kuva-analyysiä ei voitu diplomityön aikataulun puitteissa toteuttaa. Sen sijaan kuvien esikäsittelyn, tallennuksen ja visualisoinnin yhteydessä tuli esille runsaasti kuvankäsittelyä vaativia operaatioita.

Projektin mielenkiintoisuus olikin sen monipuolisuudessa. Projektissa oli kameroiden ja linssien toiminnan selvittelyn lisäksi paljon erilaista ohjelmointia, pakkausalgoritmien selvittelyä, kuvankäsittelyä ja 3D grafiikkaa vaativia töitä. Lisäksi oli välillä virkistävää käydä kokoamassa kamerajärjestelmä ja tarkastamassa sen toiminta ONKALO:ssa.

LÄHTEET

[1] P. Aalto, I. Aaltonen, H. Ahokas, J. Andersson, M. Hakala, P. Hellä, J. Hudson, E. Johansson, K. Kemppainen, L. Koskinen, M. Laaksoharju, M. Lahti, S. Lindgren, A. Mustonen, K. Pedersen, P. Pitkänen, A. Poteri, M. Snellman, M. Ylä-Mella. Programme for Repository Host Rock Characterization in the ONKALO (ReRoc). Eurajoki. Working Report 2009-31. 64 s.

[2] M. Hakala, J. A. Hudson, J. P. Harrison, E. Johansson. Assessments of the potential for Rock Spalling at the Olkiluoto site. Eurajoki. Working Report 2008-83. 60 s.

[3] Olkiluoto Modelling Task Force, Rock Mechanics Group. 2009. Outline of Posiva's Olkiluoto Spalling Experiment (POSE). Sisäinen raportti. 25 s.

[4] Microsoft Corporation. HD Photo Feature specification. 2006. Saatavissa: http://www.microsoft.com/downloads/en/details.aspx?FamilyID=6fe1ebac-c7b3-4768-90bc-13d330d5ec02&displaylang=en.

[5] Hecht, Eugene. Optics 3rd ed. New York 1998. Addison Wesley Longman, Inc. 694 p.

[6] Gonzales, R. Woods, R. Digital Image Processing. Upper Saddle River, New Jersey. 2008. Pearson Education, Inc. 947 p.

[7] Jain, Anil K. Fundamentals of Digital Image Processing. Englewood Cliffs, New Jersey. 1989. Prentice-Hall, Inc. 569 p.

[8] Cass Everitt. Projective Texture Mapping. 2001. Saatavissa: http://developer.nvidia.com/object/Projective_Texture_Mapping.html.

[9] OpenMP Architecture Review Board. OpenMP Application Program Interface Version 3.0. 2008. Saatavissa: http://www.openmp.org/mp-documents/spec30.pdf.